ResiStay

Session 2023 – 2027

Submitted by:

ESA IRFAN 2023-CS-104

Supervised by:

Mr Laeeq Khan Niazi

Mam Maida Shahid

Course:

CSC-103 Object Oriented Programming

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

1.1 About ResiStay

The **ResiStay** is a user-friendly platform with distinct logins for students, resident tutors, and senior wardens. It streamlines hostel operations by allowing students to manage profiles, request leave, and submit complaints. Resident tutors oversee students in their respective hostels, while senior wardens coordinate across all hostels, ensuring efficient management.

1.1.1 Why ResiStay?

The ResiStay is a preferred choice due to its user-friendly interface, which simplifies administrative tasks, such as room assignments, leave requests, and complaint submissions. It streamlines processes, saving time and reducing manual effort for students and staff. Effective communication through notifications and updates keeps all stakeholders informed. Senior wardens benefit from centralized control over multiple hostels, facilitating efficient management. The app contributes to an enhanced hostel experience by offering a structured and organized platform.

1.1.2 Expectations from ResiStay

Users can expect an efficient and user-friendly experience with the Hostel Management App. The platform simplifies various administrative tasks, such as room assignments, leave requests, and complaint submissions, reducing manual work and saving time. Effective communication is facilitated through timely notifications and updates, ensuring that students, resident tutors, and senior wardens are well-informed. With centralized control, senior wardens can oversee multiple hostels, contributing to more organized and effective management. The app's structured and organized platform enhances the overall hostel experience for all users. Additionally, access to student data, leave records, and complaint history simplifies issue resolution and trend monitoring, promoting transparency and a smoother living environment.

1.1.3 Contribution towards CS

ResiStay make substantial contributions to the field of computer science by incorporating various principles and techniques. These applications leverage optimization algorithms for efficient room allocation, a fundamental aspect of computer science that involves balancing multiple factors to arrive at the most optimal solution. Additionally, the integration of robust security measures, such as encryption and secure authentication processes, addresses the critical domain of cybersecurity, emphasizing the importance of safeguarding personal data within these systems. The development of user-friendly interfaces within hostel apps follows principles of Human-Computer Interaction (HCI) and user experience design, fostering advancements in the application of HCI methodologies. These apps also play a role in database management, handling extensive student information and transaction records, contributing to the broader field of database systems and data modeling. The use of machine learning algorithms for predictive analysis further extends their impact, providing insights into student behavior and resource utilization, thereby contributing to the evolving landscape of artificial intelligence and machine learning in computer science.

1.2 User Types on ResiStay

ResiStay provides different access for different type of users. The users are divided into three categories, Students, Resident tutor and Senior Warden (Admin). Each user type has access to different types of command related to their need and requirements. User can login and verify their type by inputting the issued username and password for authentication. Furthermore, each client type user has different database connected to it. So that data is stored and kept for specific person.

The hierarchy and functionality of user types is as under: -

1.2.1 Warden (Owner)

The Warden Profile functionality encompasses a comprehensive suite of features, allowing wardens to efficiently oversee hostel operations. Warden can add Hostels or remove them. Wardens can check the availability of students, allot rooms, adding room, manage room occupancy, and access detailed student data. Wardens can also take actions such as removing students or resident tutors, adding new resident tutors, making announcements, displaying and managing rules and logging out. This multifaceted functionality equips wardens with the necessary tools to maintain order, address issues promptly, and ensure the overall well-being of the hostel community.

1.2.2 Resident Tutor

The Resident Tutor (RT) Profile module is designed to empower RTs with tools for effective hostel management. RTs can conveniently view the list of students assigned to them, set and enforce rules Furthermore, the RT can stay informed about announcements, hostel events, and rules, complains.This feature-rich functionality aims to streamline the RT's responsibilities and foster a positive and organized living environment.

1.2.3 Student

The Student Profile functionality caters to the diverse needs of hostel residents. Students can effortlessly navigate through options such as checking their profile details, submitting complaints or requests for issue resolution, reviewing and understanding hostel rules, and staying updated on notices and announcements. Additionally, students have quick access to change their passwords securely, leave rooms and log out seamlessly, promoting a user-centric and interactive platform for managing various facets of hostel life.

1.3 Functional Requirements of Resistay

Some of the functional requirements expected from Resistay are as under

User Type	Required Functions to be performed	Result of Action Performed
1 – Warden	Manage Hostels	He can add or remove the hostel of his own choice.
	Allot Student	Allot a room to the students in their respective hostels according to their gender.
	Manage Rooms	He can add or remove the room of his own choice.
	Manage Annoucements	Set the announcements to resident tutors and students as well as he can remove announcements
	Manage rules	Set the rules for resident tutors and students as well as he can remove announcements
	Add Rt	He can add Rt and assign him hostel according to his respective gender.
	Remove RT and Student	Remove the rt and student and his respective data from all assosciated classes.
	Log out	Warden can log out from his menu and going back to login Menu.

2-Resident Tutor	View assigned students	Display all the students that are assigned tohis hostel.
	Manage rules	Set the rules for resident tutors and students as well as he can remove announcements
	Leave Hostel	Leave Hostel and his all associated Data is deleted.
	Check announcements	Display the announcements sets by both warden and Rt.
	Check complains	Display complains from only those students which are assigned to his hostel
	Log out	Rt can log out from his menu and going back to login Menu.
3-Student	Check Profile	Student can check his profile which includes about his details of room and hostel.
	Manage Complains	Set the Complains to resident tutors As well as he can remove announcements
	Check announcements	Display the announcements sets by Warden.
	Check rules	Display the rules that are implemented by both warden and Rt.

3-Student	Leave Room	Leave Room and his all associated Data is deleted.
	Change password	Change the password after verification of old password.
	Log out	Student can log out from his menu and going back to login Menu.

1.4 Comparison with Procedural Programming:

Aspect	Procedural Programming	Object-Oriented Programming	Advantages of OOP
Program Structure	Programs are structured around procedures/functions	Programs are structured around objects and classes	OOP provides a more modular and organized approach to software development. Objects encapsulate data and behavior, leading to better code organization, reusability, and maintainability.
Data Abstraction	Data and procedures are separate entities	Data and behavior are combined into objects	OOP promotes data abstraction, allowing complex data structures to be represented by objects with well-defined interfaces. This abstraction hides the internal details of objects, making code more manageable and reducing complexity.

Aspect	Procedural Programming	Object-Oriented Programming	Advantages of OOP
Encapsulation	Limited data hiding and encapsulation capabilities	Strong encapsulation through classes and access modifiers	OOP supports encapsulation, allowing data to be hidden and accessed only through well-defined interfaces. This enhances security and data integrity, prevents unauthorized access, and enables better control over data and its manipulation.
Inheritance	No inherent support for inheritance	Supports inheritance, enabling code reuse and hierarchy	Inheritance allows classes to inherit properties and behavior from other classes, promoting code reuse, extensibility, and the creation of hierarchical relationships. It helps avoid code duplication and allows for efficient updates and maintenance.
Polymorphism	Limited or no support for polymorphism	Supports polymorphism through inheritance and interfaces	Polymorphism allows objects of different types to be treated interchangeably. It improves code flexibility, extensibility, and modularity by enabling the use of generic interfaces and abstract classes. Polymorphism simplifies code by allowing common functionality to be implemented once and reused across different objects and types.

Aspect	Procedural Programming	Object-Oriented Programming	Advantages of OOP
Modifiability and Scalability	Code changes can be complex and error-prone	Code changes are localized and more manageable	OOP facilitates code modifiability and scalability. Changes can be made more easily and are confined to specific classes or objects, reducing the impact on the overall codebase. This promotes code maintenance, extensibility, and the ability to adapt to evolving requirements or add new features without major disruptions to the existing code.
Code Reusability	Code reuse is limited and achieved through functions/procedures	Promotes code reuse through classes, inheritance, and composition	OOP encourages code reuse by providing mechanisms like inheritance, polymorphism, and composition. This leads to more efficient development, reduced redundancy, improved productivity, and easier maintenance.

1.5 OOP Concepts

1.5.1 Inheritence

In this project, There is a parent class or a base class called “Resident” and the classes “Student” (which contains the seperate information of the student) , “RT” (which contains the data of the Rt) are the child classes or derived classes and are inherited from the base class “Resident”.

1.5.2 Association

The classes are associated with each other in such a way that

- ➢ The relation between class “Hostel” and class “Room” is Composition because if there will be no hostel, there will be no rooms.

- The relation between class “Hostel” and class “Student” is Composition because if there will be no hostel, there will be no students.

- The relation between class “Room” and class “Student” is Composition because if there will be no room, there will be no student.

- The relation between class “Hostel” and class “RT” is Composition because if there will be no hostel, there will be no rt.

- The relation between class “Student” and class “Complain” is Composition because if there will be no Student, there will be no complains.

- The relation between the class “Student” and the class “Hostel” is Aggregation because without student hostel can exist.

- The relation between the classes “Society and “Admin” is also composition because in this project, Admin is the Society Admin and if society will not exist, it’s admin will not exist.

1.5.3 Polymorphism

In this Project, Static Polymorphism is used in different type of consutructors of Classes. Moreover, it is also used in Dl layer with database in selecting same type of things but by different parameters.

1.5.4 Encapsulation

In this Project, Encapsulation is used in all Bl Classes and some Dl classes also by making their attributes private.

1.6 Classes

There are total eight classes which have been implemented in this Project.

2 Resident
3 Student
4 RT
5 Hostel
6 Room
7 Complain
8 Annoucement
9 Rule

1.7(a) Database Tables

There are total eight tablesof Dtabase named as **Resistay** which have been used in this Project for insertion, retrieving, Updation and Deletion of Data.

10 Resident
11 Student
12 RT
13 Hostel
14 Room
15 Complain
16 Annoucement
17 Rule

1.7(b) SQL Queries

Following are the queries which have been used to perform on Database Tables

1. INSERT INTO Resident (Username, Password, Age, Gender, Cnic, Role) VALUES ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}')
2. DELETE FROM Resident WHERE HostelName = '{0}'
3. SELECT * FROM Resident WHERE Username = '{0}' AND Password = '{1}'
4. UPDATE Resident SET Password = '{0}' WHERE Username = '{1}'
5. SELECT ResidentId FROM Resident WHERE Username = '{0}
6. UPDATE Resident SET HostelName = '{0}' WHERE Username = '{1}'
7. DELETE FROM Resident WHERE Username = '{0}'
8. SELECT Gender FROM Resident WHERE Username = '{0}'
9. INSERT INTO Hostel (HostelName, HostelType, HostelStatus) VALUES ('{0}', '{1}', '{2}');
10. DELETE FROM Hostel WHERE HostelName = '{0}';
11. SELECT * FROM Hostel;
12. SELECT HostelName FROM Hostel;
13. SELECT HostelName FROM Hostel WHERE HostelType = '{0}';
14. SELECT * FROM Hostel WHERE HostelName = '{0}';
15. SELECT HostelName FROM Room WHERE RoomNumber = '{0}';
16. SELECT HostelName FROM Hostel WHERE HostelName = '{0}';
17. UPDATE Hostel SET HostelStatus = 'Unchecked' WHERE HostelName = '{0}';
18. UPDATE Hostel SET HostelStatus = 'Checked' WHERE HostelName = '{0}';
19. INSERT INTO Room (RoomNumber, HostelName, RoomStatus) VALUES ('{0}', '{1}', '{2}');
20. SELECT RoomNumber FROM Room WHERE HostelName = '{0}' AND RoomStatus = 'Vacant';
21. SELECT * FROM Room WHERE RoomStatus = 'Vacant';
22. DELETE FROM Room WHERE HostelName = '{0}';
23. SELECT * FROM Room WHERE RoomNumber = '{0}';

24. DELETE FROM Room WHERE RoomNumber = '{0}' AND HostelName = '{1}';
25. SELECT RoomNumber FROM Room;
26. UPDATE Room SET RoomStatus = 'Vacant' WHERE RoomNumber = '{0}' AND HostelName = '{1}';
27. UPDATE Room SET RoomStatus = 'Alloted' WHERE RoomNumber = '{0}' AND HostelName = '{1}';
28. SELECT RoomNumber FROM Room WHERE HostelName = '{0}' AND RoomNumber = '{1}';
29. INSERT INTO Student (Username, Password, Age, Gender, Cnic, Role, ResidentId, StudentStatus) VALUES ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}', '{7}');
30. DELETE FROM Student WHERE HostelName = '{0}';
31. SELECT Username FROM Student WHERE StudentStatus = 'Unalloted';
32. SELECT HostelName FROM Student WHERE Username = '{0}';
33. SELECT Username FROM Student WHERE Username = '{0}';
34. SELECT * FROM Student WHERE Username = '{0}';
35. SELECT RoomNumber FROM Student WHERE Username = '{0}';
36. DELETE FROM Student WHERE RoomNumber = '{0}';
37. UPDATE Student SET StudentStatus = 'Alloted', RoomNumber = '{0}', HostelName = '{1}' WHERE Username = '{2}';
38. UPDATE Student SET Password = '{0}' WHERE Username = '{1}';
39. SELECT * FROM Student WHERE StudentStatus = 'alloted';
40. SELECT * FROM Student WHERE HostelName = '{0}';
41. SELECT * FROM Student;
42. DELETE FROM Student WHERE Username = '{0}';
43. SELECT Username FROM Student;
44. SELECT Username FROM Student WHERE Hostelname = '{0}';
45. INSERT INTO Annoucement (Details, Date) VALUES ('{annoucement.GetDescription()}', '{annoucement.GetDate()}')
46. DELETE FROM Annoucement WHERE AnnoucemntId = {announcementId}
47. SELECT * FROM Annoucement
48. SELECT AnnoucemntId FROM Annoucement
49. INSERT INTO Complain (Detail, Date, StudentName) VALUES ('{complain.GetDescription()}', '{complain.GetDate()}', '{username}')
50. DELETE FROM Complain WHERE ComplainId = {complainId}
51. SELECT * FROM Complain WHERE StudentName = '{studentname}'
52. SELECT ComplainId FROM Complain WHERE StudentName = '{studentName}'
53. DELETE FROM Complain WHERE StudentName = '{studentName}'
54. DELETE FROM Rt WHERE HostelName = '{hostelName}'
55. INSERT INTO Rt (Username, Password, Age, Gender, Cnic, Role, PhoneNo, ResidentId, HostelName) VALUES ('{resident.GetName()}', '{resident.GetPassword()}', '{resident.GetAge()}', '{resident.GetGender()}', '{resident.GetCnic()}', '{resident.GetRole()}', '{resident.GetPhoneNumber()}', {residentId}, '{resident.GetHostel().GetHostelName()}')

56. DELETE FROM Rt WHERE Username = '{rtName}'
57. SELECT HostelName FROM Rt WHERE HostelName = '{username}'
58. SELECT Username FROM Rt WHERE Username = '{username}'
59. UPDATE Rt SET Password = '{password}' WHERE Username = '{name}'
60. SELECT * FROM Rt WHERE Username = '{name}'
61. SELECT * FROM Rt
62. SELECT Username FROM Rt
63. INSERT INTO Rules (Description, Date) VALUES ('{rule.GetDescription()}', '{rule.GetDate()}')
64. DELETE FROM Rules WHERE RulesId = {ruleId}
65. SELECT * FROM Rules
66. SELECT RulesId FROM Rules

1.8 Design Pattern Implementation

- **BL (Business Layer)**

This project utilizes the BL Design Pattern in such a way that all the BL classes contains the Business logic funtions and all the variables are declared in it and all getters () and setters () functions are also included in this layer.

- **DL (Data Layer with file handling)**

This project utilizes the DL Design Pattern in such a way that the DL with File Handling classes contain all the Lists, their functions and all functions related to Lists like AddingDataToList () etc. and all the CRUD functions of classes. Moreover, it also includes function related with files

- **DL (Data Layer with database)**

This project utilizes the DL Design Pattern in such a way that the DL with Database classes contain all the Lists, their functions and all functions related to Table like AddingDataToList () etc. and all the CRUD functions of classes. Moreover, it also includes function related with database tables

- **DL Interfaces**

In this folder, we can write functions names only by which we can switch to File Handling and Database

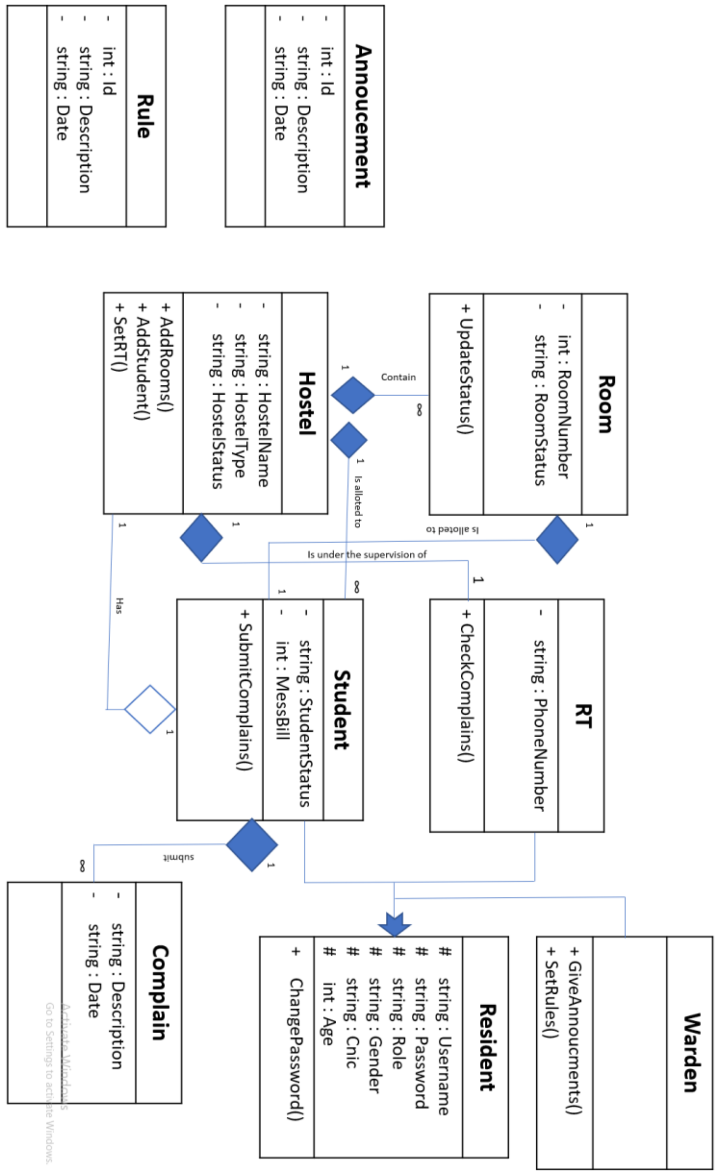
- **Utilities**

This includes all fuction related to validations

- **UI (User Interface Layer)**

This project utilizes the UI Design Pattern in such a way that all the forms and functions related with User interfaces in Console are included in UI classes.

1.9 CRC Card



1.10 Wireframes:

The following is the wireframe of ResiStay Displayed in Command Line Interface:

1.10.1 Basic Features (For All types of Users):

1.10.1.1 Startup Face:

Startup interface is filled with animations to provide a premium experience to user about using the application.

1.10.1.1.1 Main Header

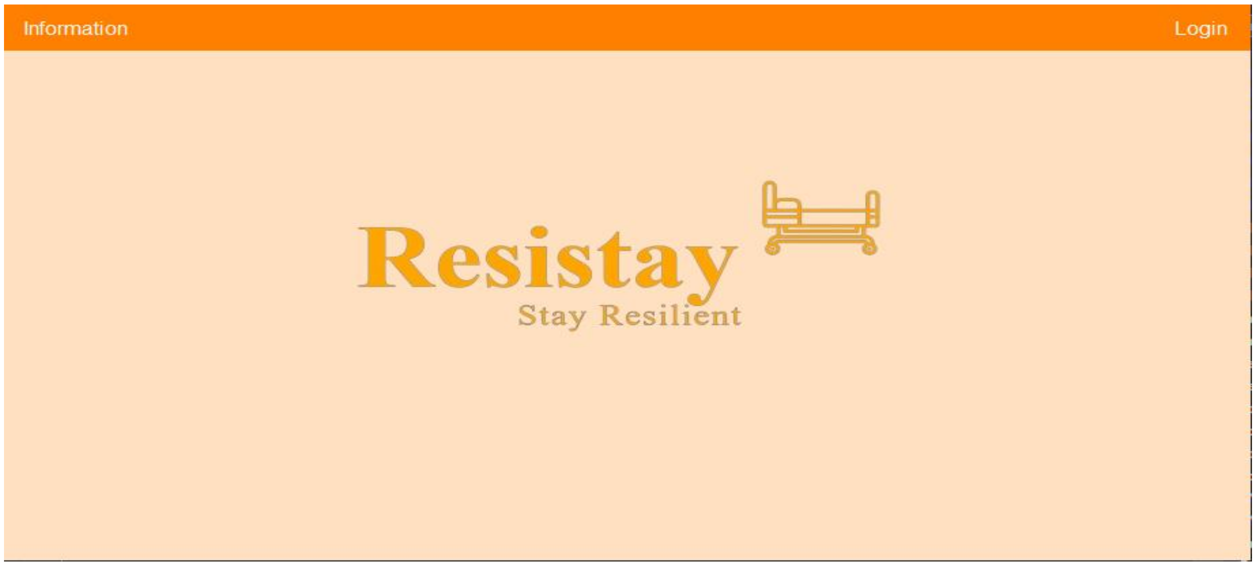
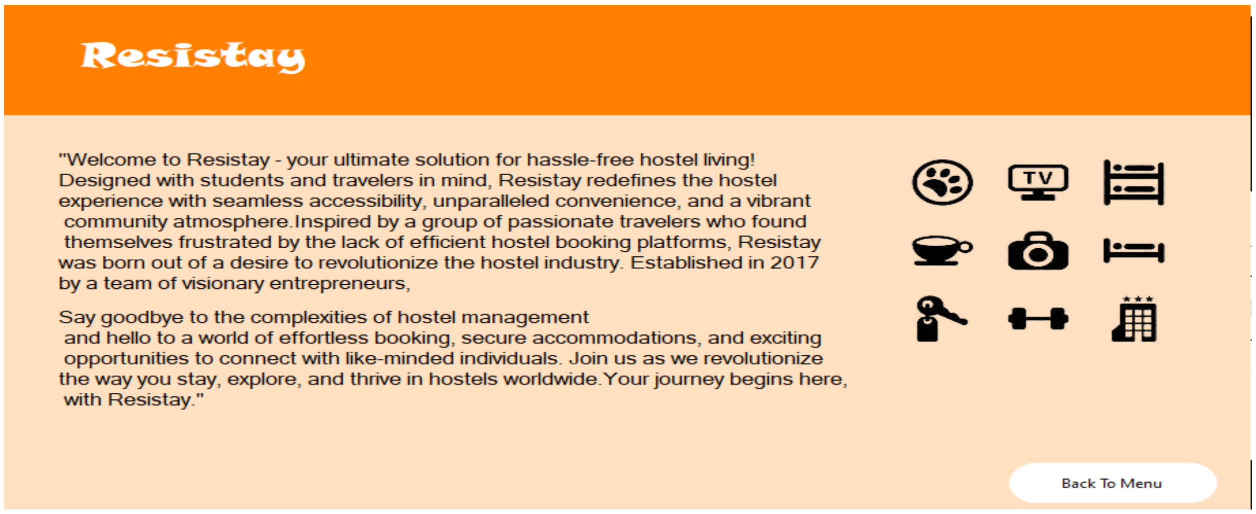


Figure 1 Resistay Main Page

1.10.1.1.2 Resistay Information



"Welcome to Resistay - your ultimate solution for hassle-free hostel living! Designed with students and travelers in mind, Resistay redefines the hostel experience with seamless accessibility, unparalleled convenience, and a vibrant community atmosphere.Inspired by a group of passionate travelers who found themselves frustrated by the lack of efficient hostel booking platforms, Resistay was born out of a desire to revolutionize the hostel industry. Established in 2017 by a team of visionary entrepreneurs,

Say goodbye to the complexities of hostel management and hello to a world of effortless booking, secure accommodations, and exciting opportunities to connect with like-minded individuals. Join us as we revolutionize the way you stay, explore, and thrive in hostels worldwide.Your journey begins here, with Resistay."

Figure 2 Resistay Information

1.10.1.1.3 Sign In

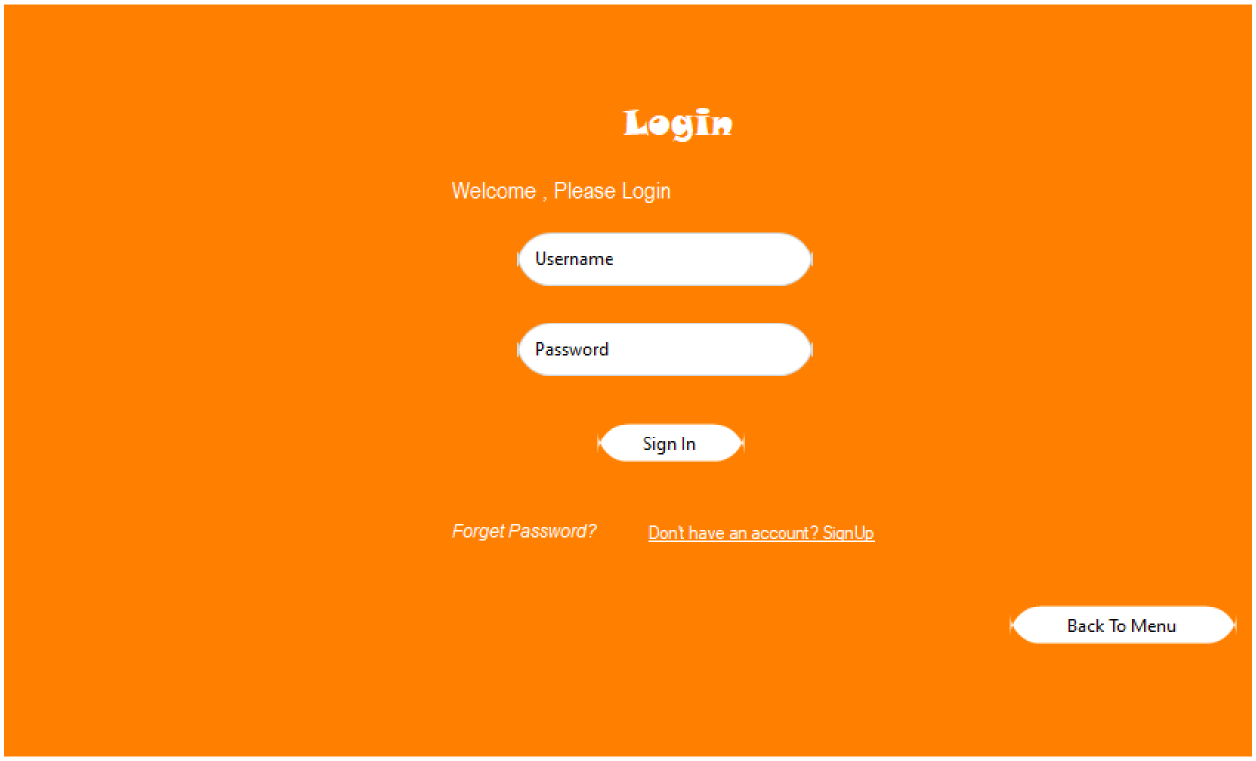


Figure 3 Sign In

1.10.1.1.3 Sign In

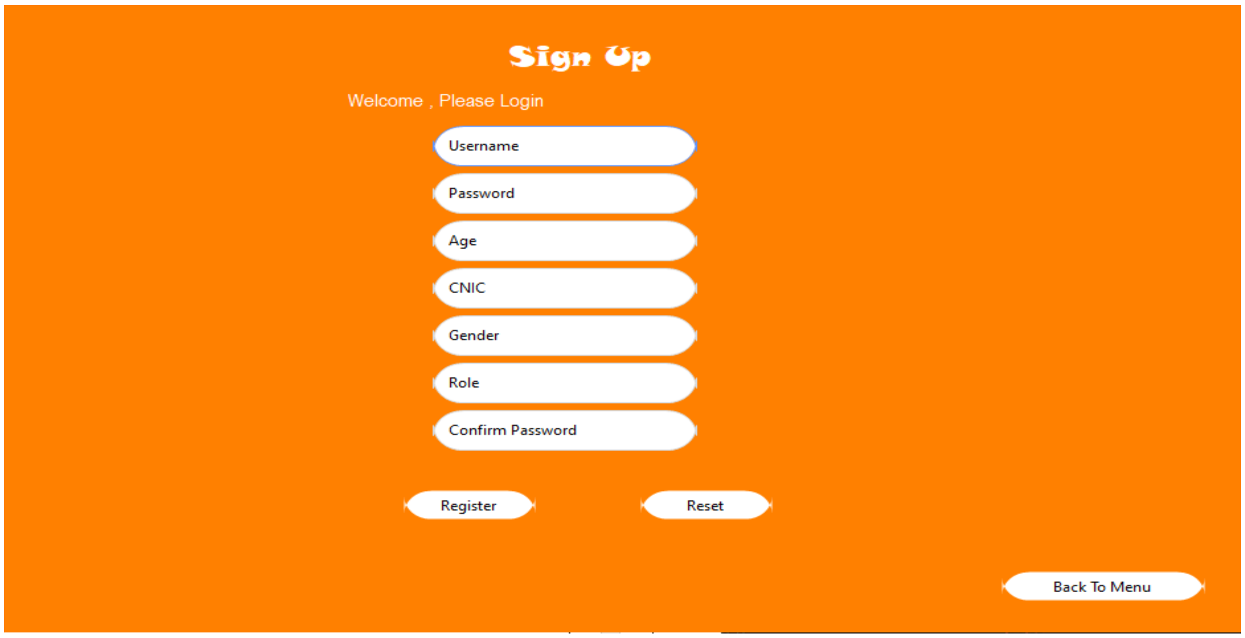


Figure 4 Sign Up

1.10.1.2 Log Out

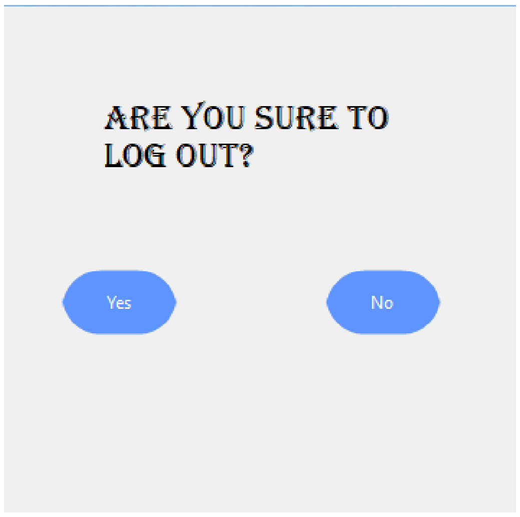


Figure 5 Log Out

1.10.2 User Type (Warden):

1.10.2.1 Warden Menu

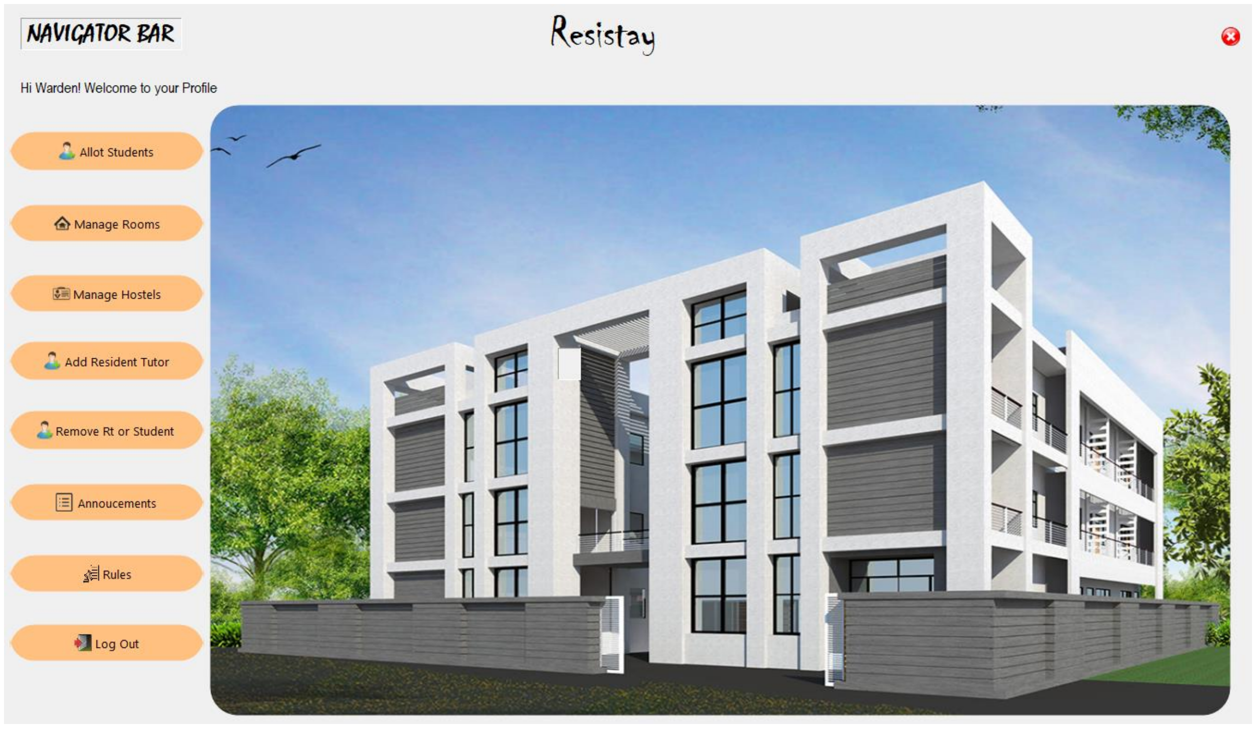
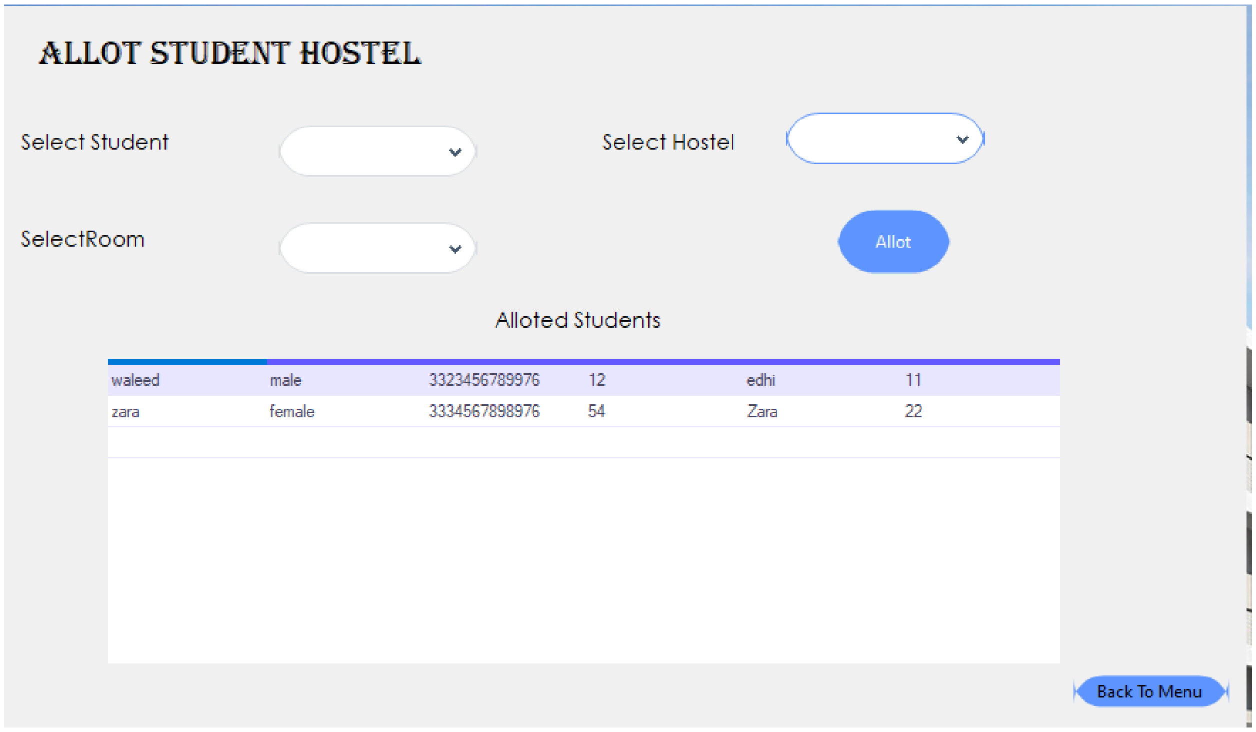


Figure 6 warden menu

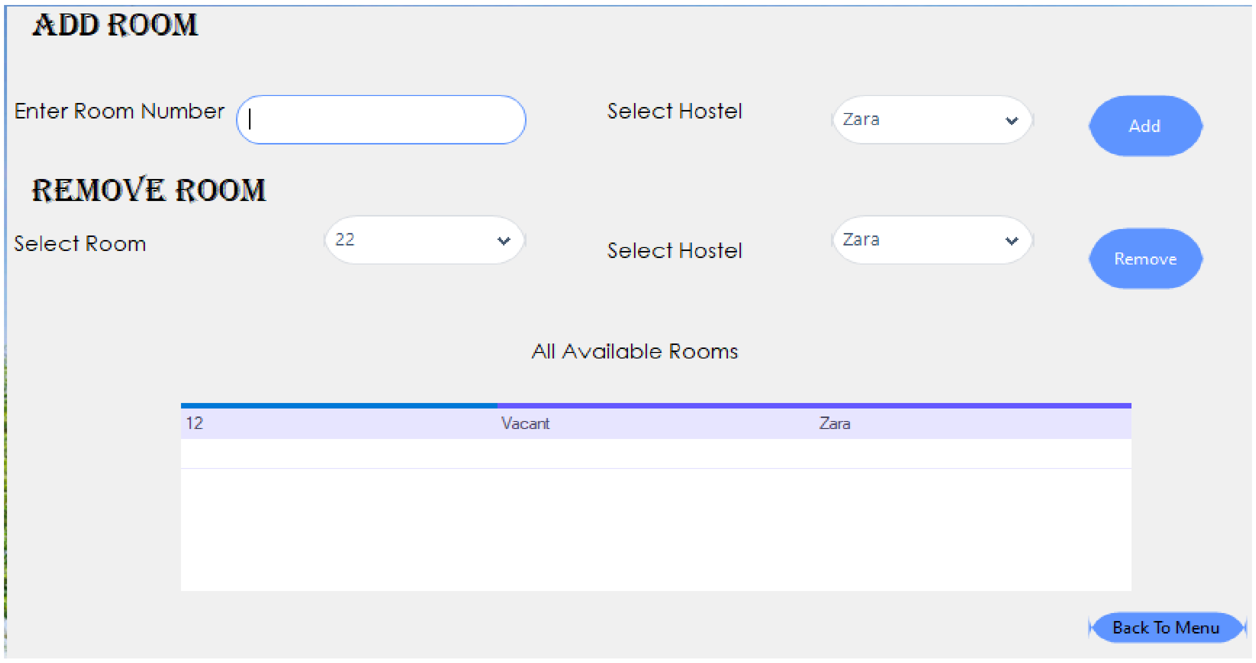
1.10.2.2 Allot Student



waleed	male	3323456789976	12	edhi	11
zara	female	3334567898976	54	Zara	22

Figure 7 Allot Student

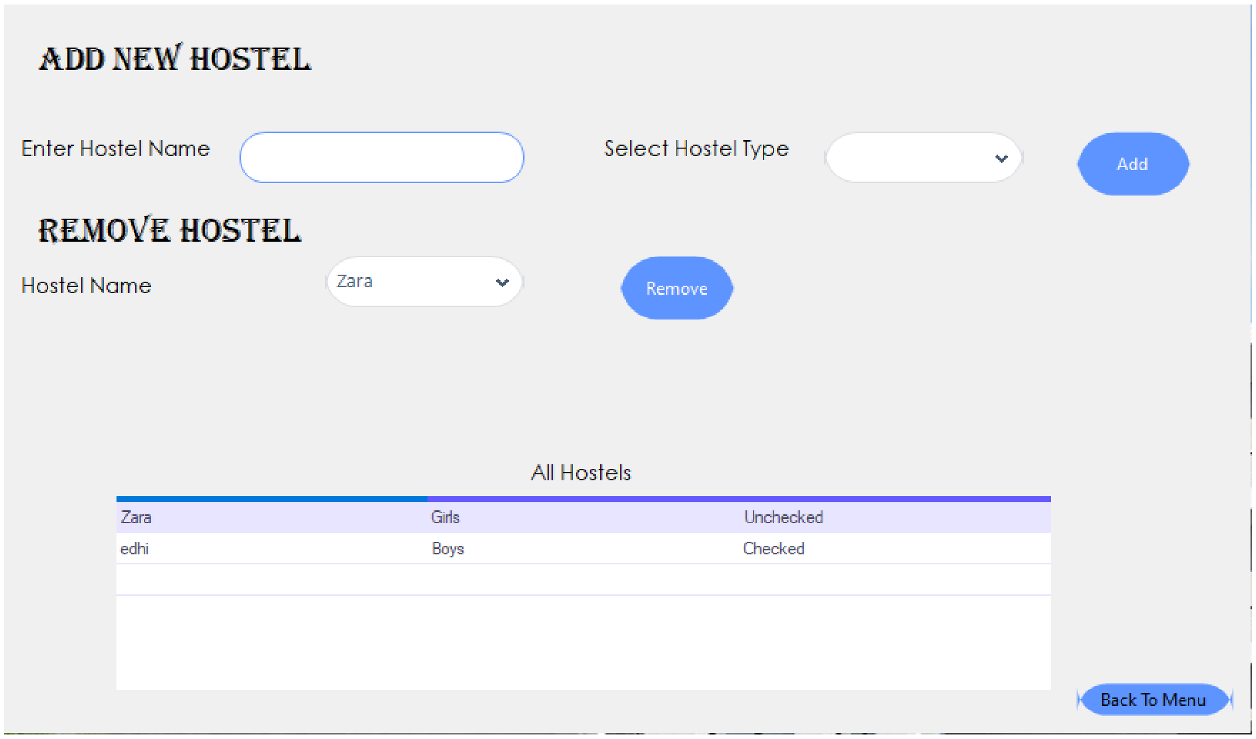
1.10.2.3 Manage Rooms



12	Vacant	Zara

Figure 8 Manage Rooms

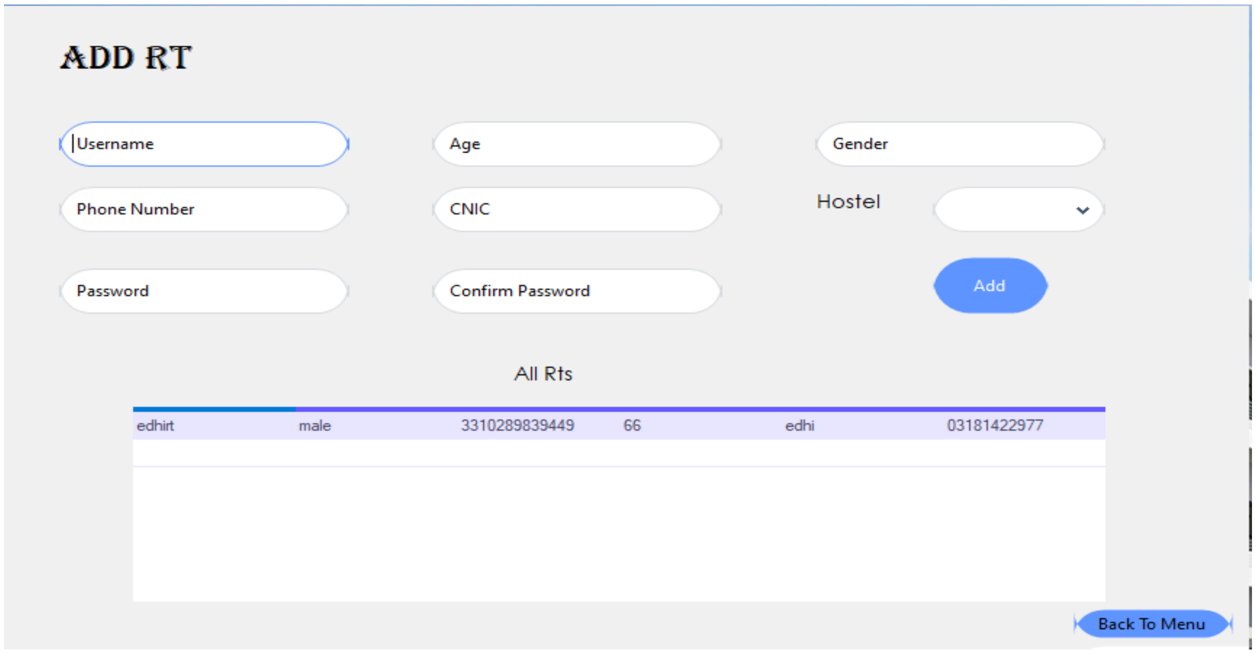
1.10.2.4 Manage Hostel



Zara	Girls	Unchecked
edhi	Boys	Checked

Figure 9 Manage Hostel

1.10.2.5 Add Resident Tutor



edhirt	male	3310289839449	66	edhi	03181422977

Figure 10 AddRT

1.10.2.6 Remove RT and Student

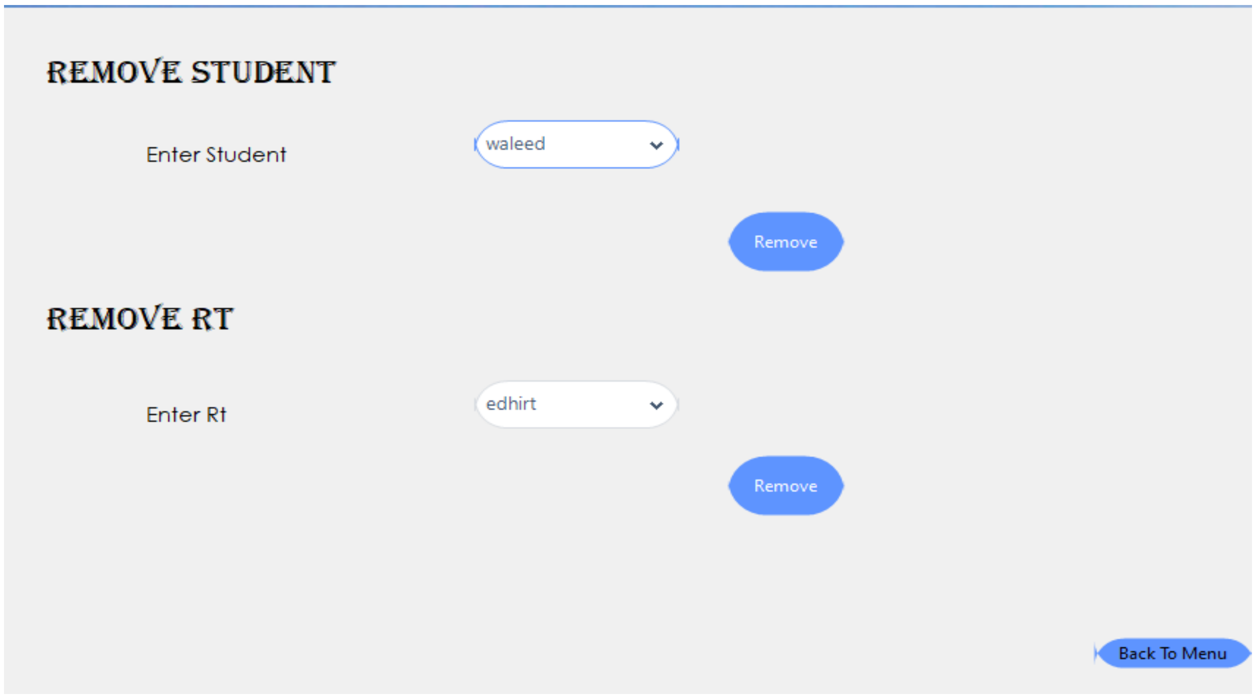
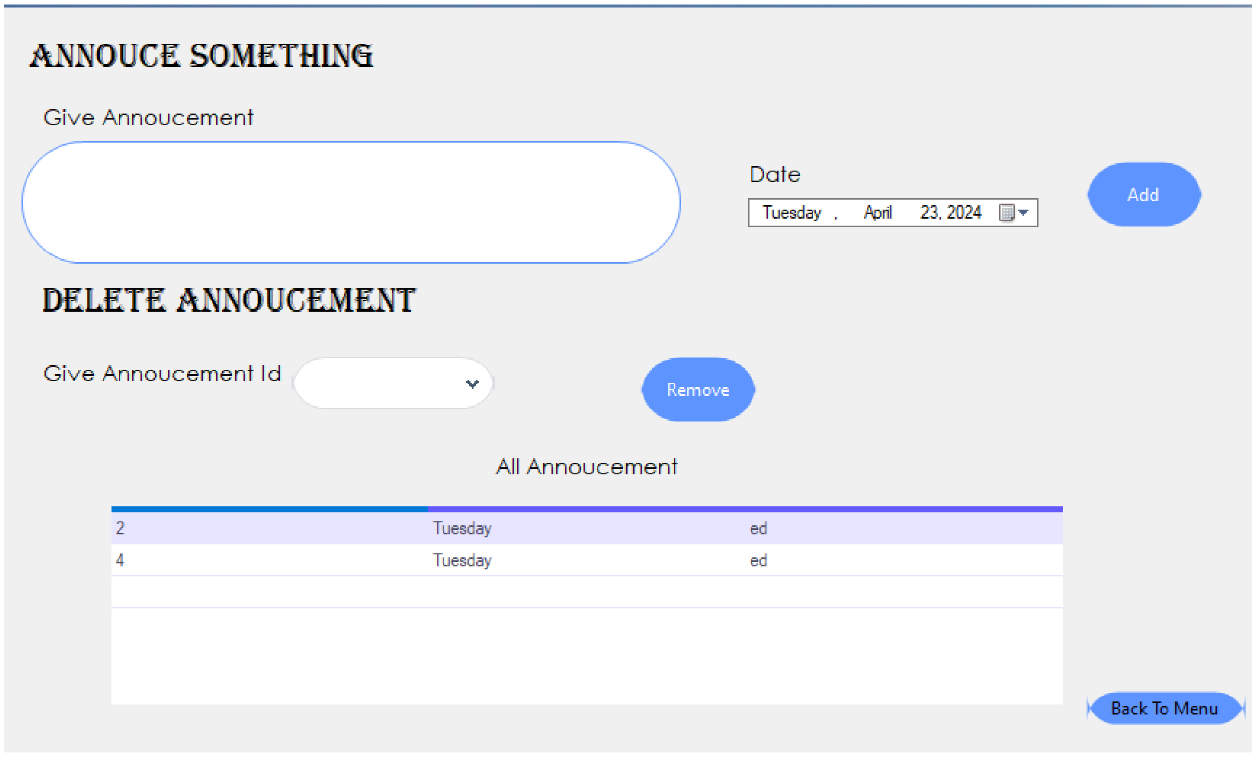


Figure 11 Remove Rt and Student

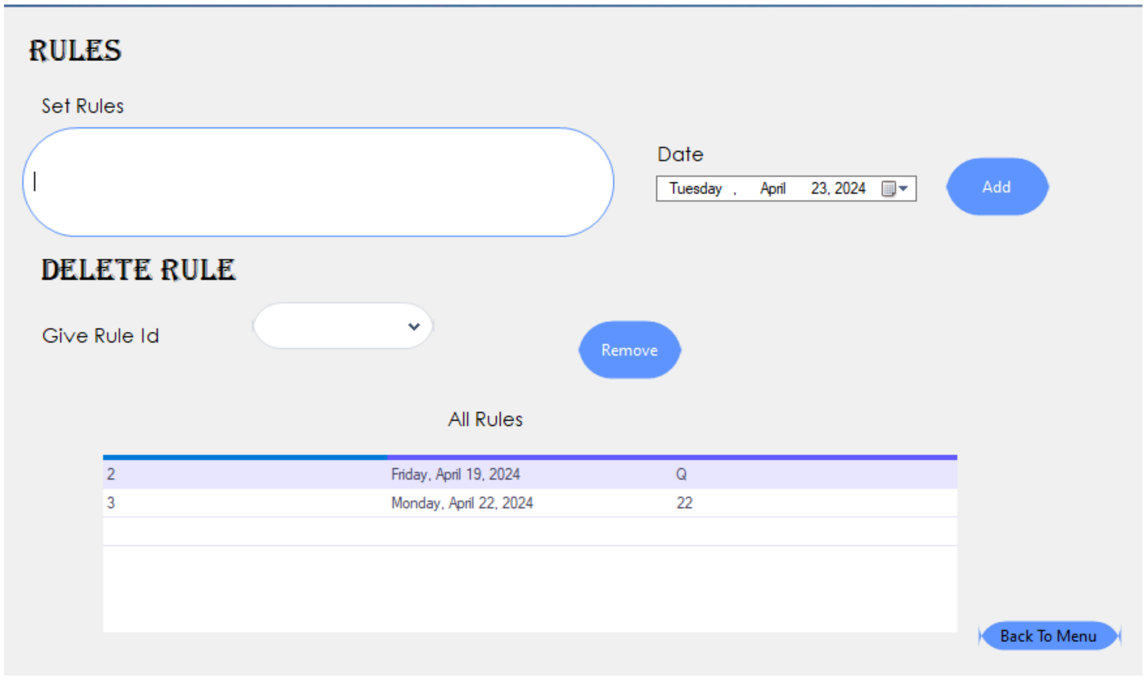
1.10.2.7 Annoucements



2	Tuesday	ed
4	Tuesday	ed

Figure 12 Annoucement

1.10.2.8 Rules



2	Friday, April 19, 2024	Q
3	Monday, April 22, 2024	22

Figure 13 Rules

1.10.3 User Type (Resident Tutor)

1.10.3.1 Rt Menu

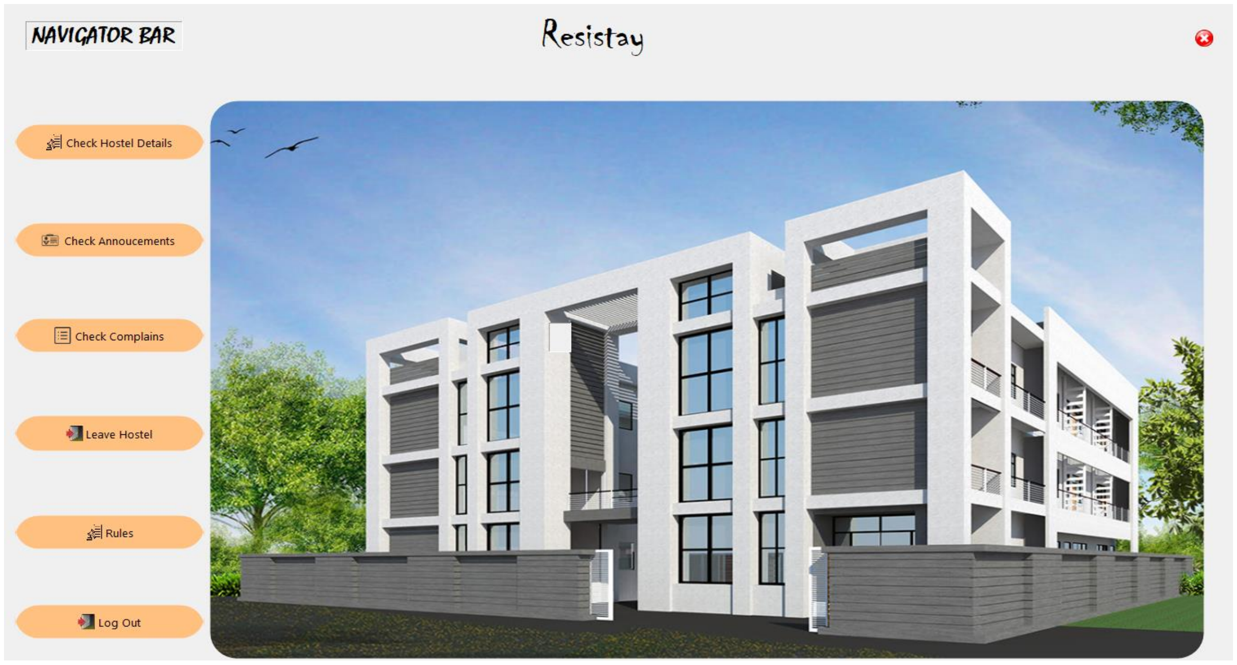
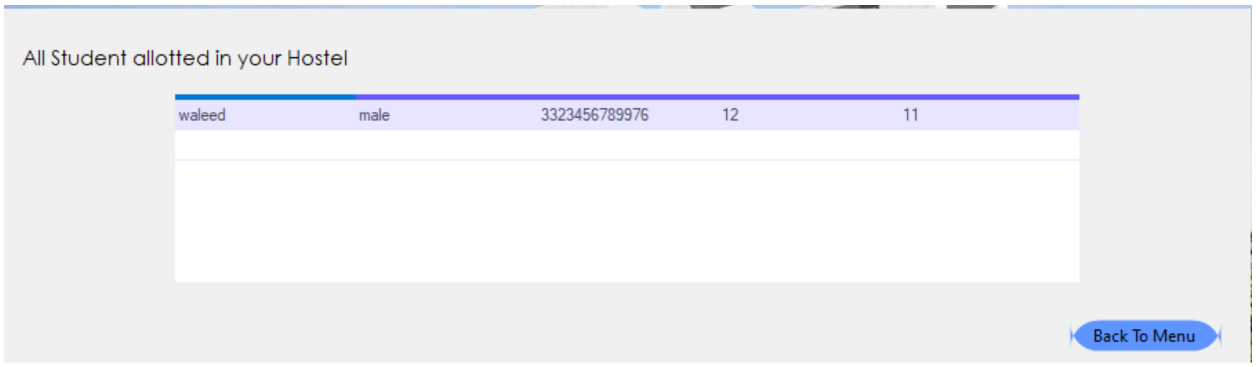


Figure 14 RT Menu

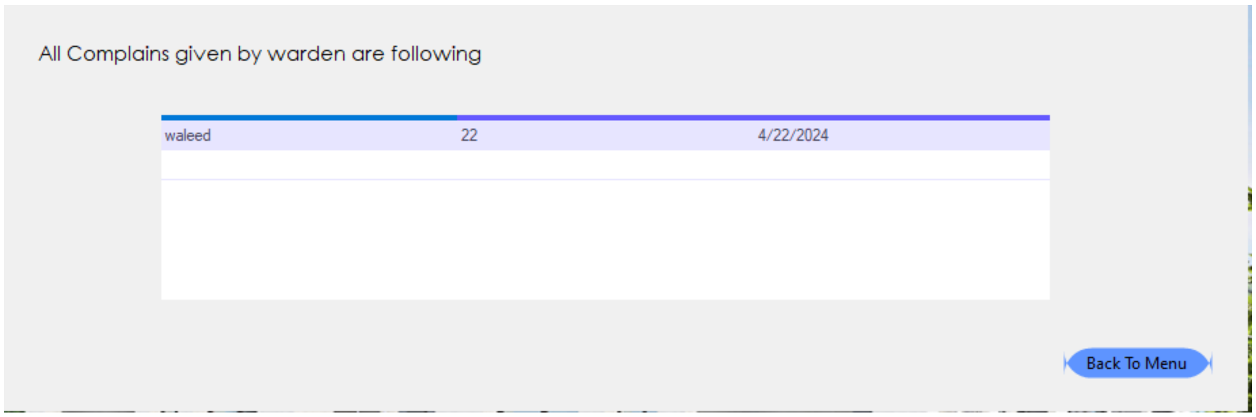
1.10.3.2 Hostel Details


All Student allotted in your Hostel

waleed	male	3323456789976	12	11

Figure 15 Hostel Details

1.10.3.3 Check Complains


All Complains given by warden are following

waleed	22	4/22/2024

Figure 16 Check Complains

1.10.3.4 Leave Hostel

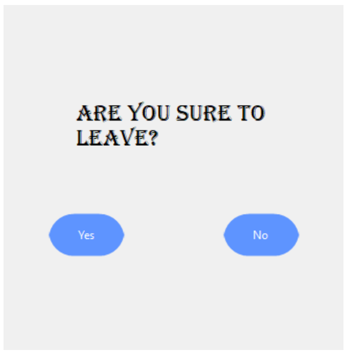

ARE YOU SURE TO LEAVE?

Figure 17 Leave

1.10.4 User Type (Student)

1.10.4.1 Student Menu

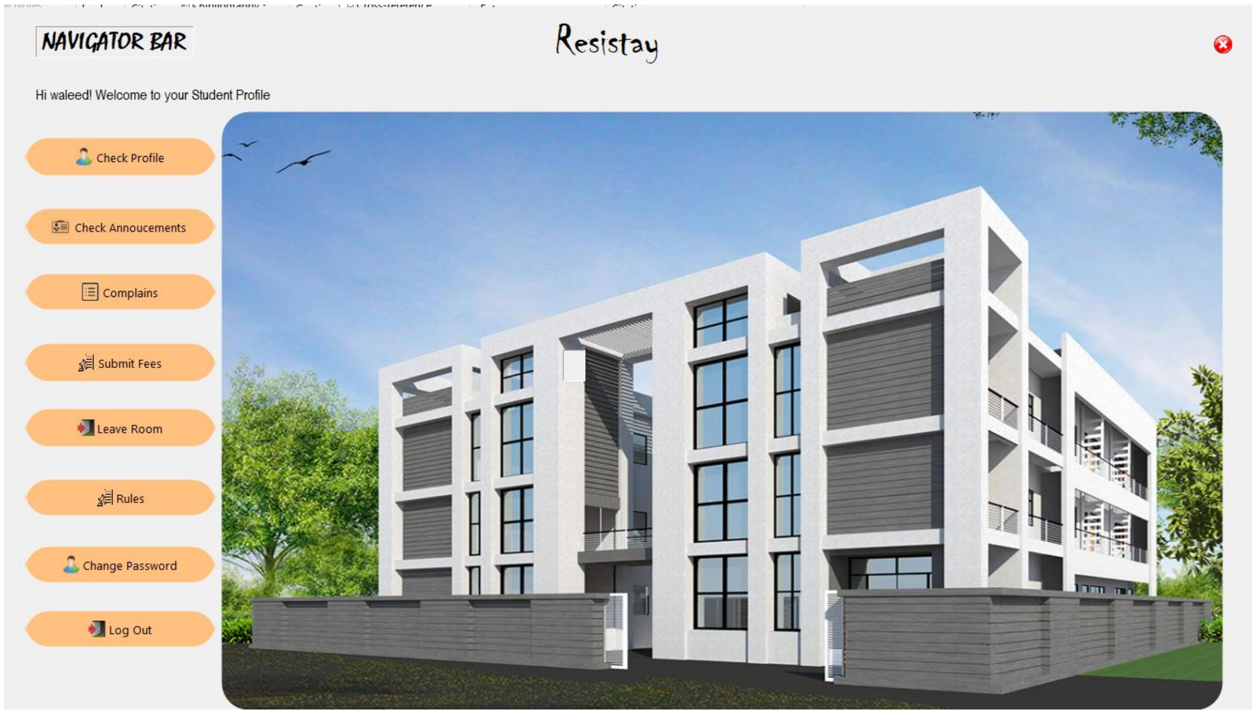
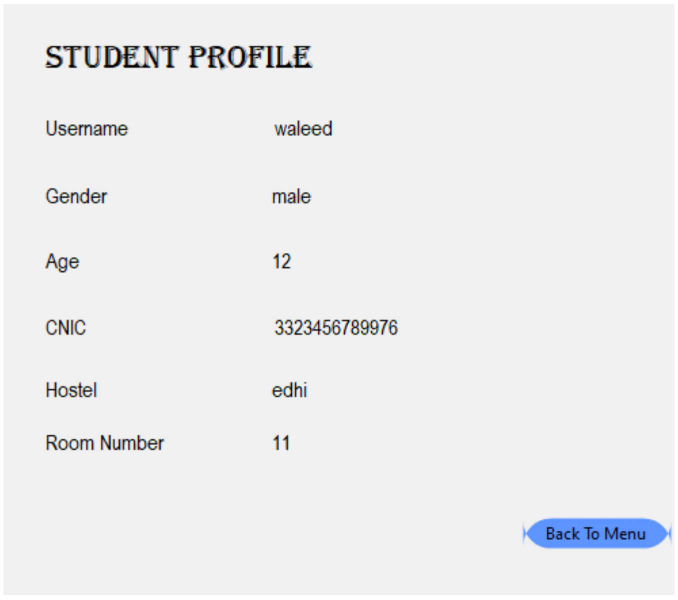


Figure 18 Student Menu

1.10.4.1 Student Profile



STUDENT PROFILE

Username	waleed
Gender	male
Age	12
CNIC	3323456789976
Hostel	edhi
Room Number	11

Figure 19 Student Profile

1.10.4.2 Complains

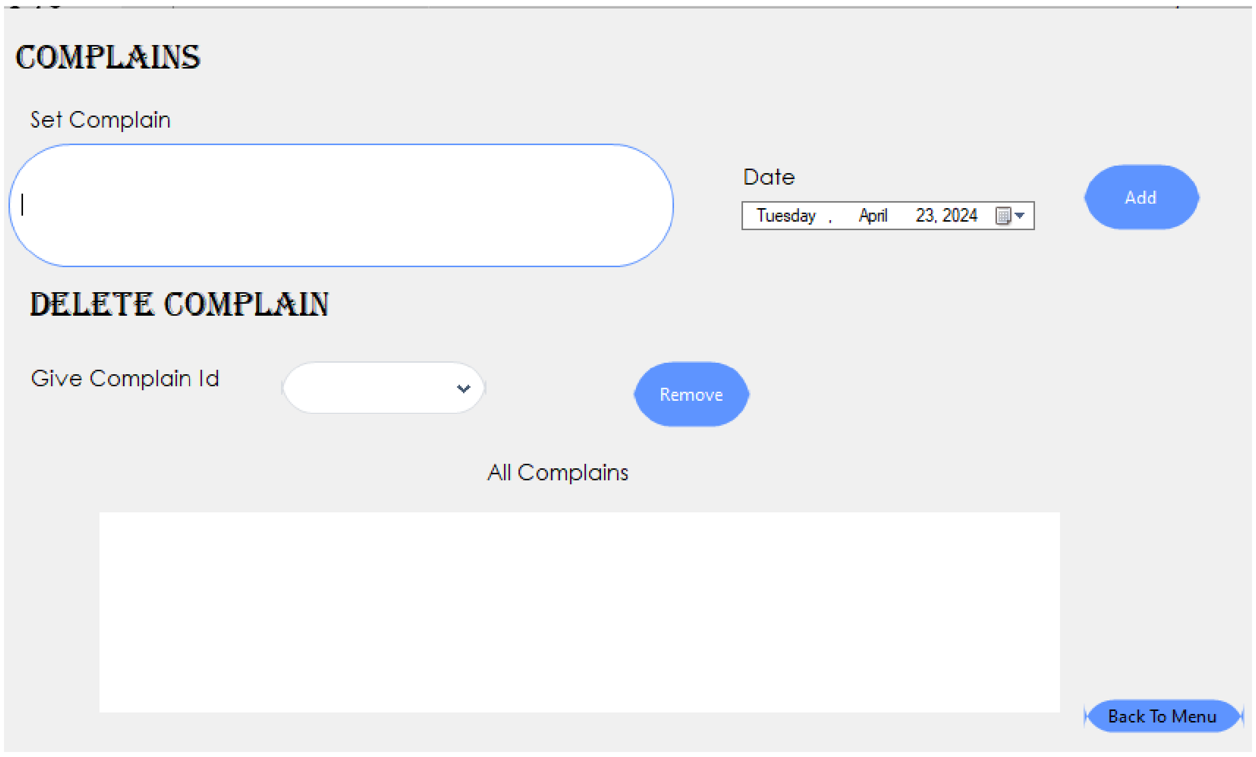


Figure 20 Complains

1.10.4.3 Change Password

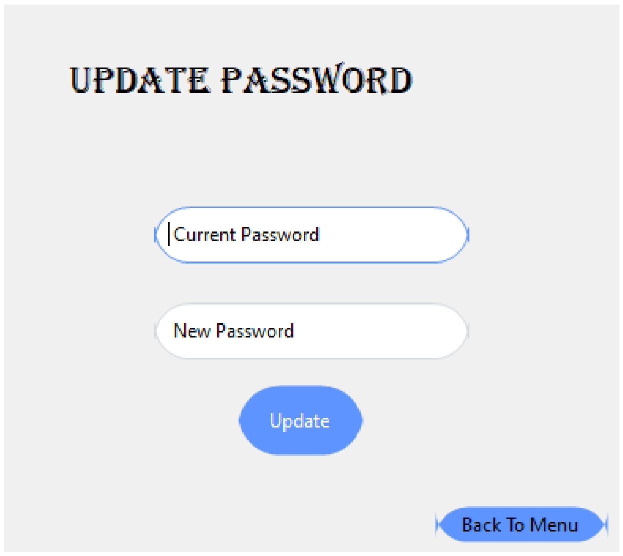


Figure 21 Change Password

1.11 Code

1.11.1 BL Folder

1.11.1.1 Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class Student: Resident
    {

        private Room Room;
        private string StudentStatus;
        List<Complain> Complains = new List<Complain>();

        public Student(string name, string password, int age, string cnic, string
role, string gender) : base(name,password,age,cnic,role,gender)
        {
            this.StudentStatus = "Unalloted";
        }
        public Student(string name, string password, int age, string cnic, string
role, string gender,Hostel hostel, Room room,string status) : base(name, password,
age, cnic, role, gender,hostel)
        {
            this.Room = room;
            this.StudentStatus= status;

        }
        public Student(string name, string password, int age, string cnic, string
role, string gender, Hostel hostel, Room room, string status, List<Complain>
complains)
            : base(name, password, age, cnic, role, gender, hostel)
        {
            this.Room = room;
            this.StudentStatus = status;
            this.Complains = complains;
        }


        public string GetStudentStatus() { return this.StudentStatus; }
        public void SetStudentStatus(string studentStatus) {  this.StudentStatus =
studentStatus; }
        public Room GetRoom() {  return this.Room; }
        public void SetRoom(Room room) { this.Room = room; }
        public List<Complain> GetComplins()
        { return this.Complains; }
        public void SetComplains(List<Complain> complains)
        {
            this.Complains = complains;
        }
        public void AddComplains(Complain complain)
        {
            Complains.Add(complain);
        }


    }
}
```

1.11.1.2 Resident.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
   public class Resident
    {
        protected string Name;
        protected string Password;
        protected int Age;
        protected string Cnic;
        protected string Role;
        protected string Gender;
        protected Hostel HostelName;

        public Resident(string name, string password, int age, string cnic, string
role, string gender)
        {
            Name = name;
            Password = password;
            Age = age;
            Cnic = cnic;
            Role = role;
            Gender = gender;


        }
        public Resident() { }
        public Resident(string name, string password, int age, string cnic, string
role, string gender, Hostel hostel ) : this(name, password, age, cnic, role,
gender)
        {
            this.HostelName = hostel;
        }
        public string GetName()
        {
            return Name;
        }

        public void SetName(string name)
        {
            Name = name;
        }

        public string GetPassword()
        {
            return Password;
        }

        public void SetPassword(string password)
        {
            Password = password;
        }

        public int GetAge()
        {
            return Age;
        }

        public void SetAge(int age)
        {
            Age = age;
        }

        public string GetCnic()
        {
            return Cnic;
        }

        public void SetCnic(string cnic)
        {
            Cnic = cnic;
        }

        public string GetRole()
        {
            return Role;
        }

        public void SetRole(string role)
        {
            Role = role;
        }

        public string GetGender()
        {
            return Gender;
        }

        public void SetGender(string gender)
        {
            Gender = gender;
        }

        public Hostel GetHostel()
        {
            return this.HostelName;
        }
        public void SetHostel(Hostel hostel)
        {
            this.HostelName = hostel;
        }


    }
}
```

1.11.1.3 Rt.cs

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class RT : Resident
    {
        private string PhoneNumber;
        public RT()
        {

        }

        public RT(string name, string password, int age, string cnic, string role,
string gender) :  base(name, password, age, cnic, role, gender)
        {
        }
        public RT(string name, string password, int age, string cnic, string role,
string gender, Hostel hostel, string phonenumber) : base(name, password, age,
cnic, role, gender, hostel)
        {
            this.PhoneNumber = phonenumber;
        }
        public string GetPhoneNumber() { return this.PhoneNumber; }
        public void SetPhoneNumber(string phoneNumber) { this.PhoneNumber =
phoneNumber; }

    }
}
```

1.11.1.4 Room.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class Room
    {
        private int Number;
        private string  RoomStatus;
        Hostel Hostel;

        public Room()
        {
        }
        public Room(int number)
        {
            this.Number = number;
            this.RoomStatus = "Vacant";
        }
        public Room (int number, string roomStatus)
        {
            this.Number = number;
            this.RoomStatus = roomStatus;
        }
        public Room(int number, string roomStatus, Hostel hostel)
        {
            this.Number = number;
            this.RoomStatus = roomStatus;
            this.Hostel = hostel;
        }
        public int GetNumber()  {  return Number; }

        public void SetNumber(int number){ this.Number = number;}

        public string GetRoomStatus()  { return RoomStatus;}

        public void SetRoomStatus(string roomStatus) { this.RoomStatus = 
roomStatus; }
        public void SetHostel(Hostel hostel) { this.Hostel = hostel;}
        public Hostel GetHostel() { return this.Hostel; }



    }
}
```

1.11.1.5 Hostel.cs

```
using System;

using System.Collections.Generic;


namespace ResistayDLL.Bl

{

    public class Hostel

    {

        private string HostelName;

        private string HostelType;

        private string HostelStatus;

        private List<Room> RoomsList = new List<Room>();

        private List<Student> StudentsList = new List<Student>();

        private RT Rt = new RT();


        public Hostel()

        {


        }


        public Hostel(string hostelName, string hostelType)

        {

            HostelName = hostelName;

            HostelType = hostelType;

            HostelStatus = "Unchecked";

        }


        public Hostel(string hostelName, string hostelType, string hostelStatus)

        {

            HostelName = hostelName;

            HostelType = hostelType;

            HostelStatus = hostelStatus;
        }

        public Hostel(string hostelName, string hostelType, string hostelStatus,
List<Room> roomsList)
        {
            HostelName = hostelName;
            HostelType = hostelType;
            HostelStatus = "Unchecked";
            RoomsList = roomsList;
        }

        public Hostel(string hostelName, string hostelType, string hostelStatus,
List<Room> roomsList, RT rt)
        {
            HostelName = hostelName;
            HostelType = hostelType;
            HostelStatus = hostelStatus;
            RoomsList = roomsList;
            Rt = rt;
        }

        public string GetHostelName()
        {
            return HostelName;
        }

        public void SetHostelName(string hostelName)
        {
            HostelName = hostelName;
        }

        public string GetHostelType()
        {
            return HostelType;
        }

        public void SetHostelType(string hostelType)
        {
            HostelType = hostelType;
        }

        public string GetHostelStatus()
        {
            return HostelStatus;
        }

        public void SetHostelStatus(string hostelStatus)
        {
            HostelStatus = hostelStatus;
        }

        public List<Room> GetRoomsList()
        {
            return RoomsList;
        }

        public void SetRoomsList(List<Room> roomsList)
        {
            RoomsList = roomsList;
        }

        public List<Student> GetStudentsList()
        {
            return StudentsList;
        }
        public void SetStudentsList(List<Student> studentsList)
        {
            StudentsList = studentsList;
        }

        public RT GetRt()
        {
            return Rt;
        }

        public void SetRt(RT rt)
        {
            Rt = rt;
        }
        public void AddRooms(int room)
        {
            RoomsList.Add(new Room(room));
        }
        public void AddStudents(Student student)
        {
            StudentsList.Add(new Student(student.GetName(), student.GetPassword(),
student.GetAge(), student.GetCnic(), student.GetRole(), student.GetGender(),
student.GetHostel(), student.GetRoom(), student.GetStudentStatus()));
        }
    }
}
```

1.11.1.6 Complain.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class Complain
    {
        private string Date;
        private string Description;

        public Complain (string Description, string date)
        {
            this.Description = Description;
            Date = date;
        }
        public Complain()
        {


        }
        public string GetDescription() { return Description; }
        public void SetDescription(string Description) { this.Description
=Description; }
        public string GetDate() { return Date; }
        public void SetDate(string Date) { this.Date = Date; }
    }
}
```

1.11.1.7 Annoucement.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class Annoucement
    {
        private string Description;
        private string Date;
        private int Id;

        public Annoucement()
        {

        }
        public Annoucement(string description,string date)
            {
            this.Description = description;
            this.Date = date;
        }
        public Annoucement(string description, string date,int Id)
        {
            this.Description = description;
            this.Date = date;
            this.Id = Id;
        }
        public string GetDescription() { return Description; }
        public void SetDescription(string description) { this.Description =
description;}
        public string GetDate() { return Date;}
        public void SetDate(string date) { }
        public int GetId() { return Id;}
        public void SetId(int id) {  this.Id = id;}

    }
}
```

1.11.1.8 Rule.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDLL.Bl
{
    public class Rule
    {
        private string Description;
        private string Date;
        private int Id;
        public Rule() { }
        public Rule(string description, string date)
        {
            Description = description;
            Date = date;
        }
        public Rule(string description, string date,int ruleId)
        {
            Description = description;
            Date = date;
            Id = ruleId;
        }


        public string GetDescription() { return Description; }

        public void SetDescription(string description) { this.Description = 
description; }

        public string GetDate() { return Date; }

        public void SetDate(string date) { this.Date = date; }


        public int GetId() { return Id; }

        public void SetId(int id) { this.Id = id; }


    }

}
```

1.11.2 DL Folder with Database

1.11.2.1 StudentDb.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


using ResistayDll.DLInterfaces;


namespace ResistayDll.DlDb

{

    public class StudentDb:IStudent
    {
        IComplain ComplainDb = new ComplainDb();
        Ihostel HostelDb = new HostelDb();
        IRoom RoomDb = new RoomDb();



        public bool AddStudent(Student resident,int ResidentId ,string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("insert into Student (Username, Password, 
Age,Gender,Cnic,Role,ResidentId,StudentStatus) VALUES('{0}', '{1}', '{2}','{3}', '{4}', '{5}','{6}','{7}')", 
resident.GetName(), resident.GetPassword(), resident.GetAge(), resident.GetGender(), 
resident.GetCnic(), resident.GetRole(),ResidentId,resident.GetStudentStatus());
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  bool DeleteStudentByHostelName(string hostelName, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            int rowsAffected = 0;

            if (connection.State != ConnectionState.Open)
                connection.Open();

            string query = string.Format("DELETE FROM Student WHERE HostelName = '{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();

            connection.Close();

            return rowsAffected > 0;
        }
        public  List<String> GetNameOfUnallotedStudents(string connectionString)
        {
            List<String> UnallotedStudents = new List<String>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = "SELECT Username FROM Student where StudentStatus = 'Unalloted'";
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();



            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);
                UnallotedStudents.Add(username);
```

```
            }


            reader.Close();

            connection.Close();


            return UnallotedStudents;


        }

        public string GetHostelOfSelectedStudent(string username, string connectionString)

        {

            string hostel = null;

            SqlConnection connection = new SqlConnection(connectionString);

            connection.Open();

            string query = string.Format("SELECT HostelName FROM Student where Username = '{0}'",
username);

            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();

            if (reader.Read())

            {

                hostel = reader.GetString(0);

            }

            reader.Close();

            connection.Close();

            return hostel;




        }

        public  bool IsDuplicateStudent(string username, string connectionString)
        {
            string resident = null;
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT Username FROM Student WHERE Username = '{0}'",
    username);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                resident = reader.GetString(0);
            }
            reader.Close();
            connection.Close();

            return resident == null;
        }
        public Student GetStudentByName(string name, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("select * from Student where Username = '{0}' ", name );
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();
            Student student;
            if (reader.Read())
            {
```

```
            string Username = Convert.ToString(reader["Username"]);

            string Password = Convert.ToString(reader["Password"]);

            string Gender = Convert.ToString(reader["Gender"]);

            int Age = Convert.ToInt32(reader["Age"]);

            string Cnic = Convert.ToString(reader["Cnic"]);

            string hostelName = Convert.ToString(reader["HostelName"]);

            string roomNumber = Convert.ToString(reader["RoomNumber"]);

            string studentStatus = Convert.ToString(reader["StudentStatus"]);

            string Role = Convert.ToString(reader["Role"]);


            student = new Student(Username, Password, Age, Cnic, Role, 
Gender,HostelDb.GetHostelByName(hostelName,connectionString),RoomDb.GetRoomByRoomNumb
er(roomNumber,connectionString),studentStatus,ComplainDb.GetAllComplainsofStudent(Username,c
onnectionString));

            return student;

          }

          else return null;

        }

        public  int GetRoomNoOfSelectedStudent(string studentName, string connectionString)

        {

          int roomNumber = -1;

          SqlConnection connection = new SqlConnection(connectionString);

          connection.Open();

          string query = string.Format("SELECT RoomNumber FROM Student where Username = '{0}'", 
studentName);

          SqlCommand command = new SqlCommand(query, connection);

          SqlDataReader reader = command.ExecuteReader();

          if (reader.Read())

          {

            roomNumber = reader.GetInt32(0);
          }
          reader.Close();
          connection.Close();
          return roomNumber;




        }
        public  bool DeleteStudentByRoomNumber(int roomNumber, string connectionString)
        {
          SqlConnection connection = new SqlConnection(connectionString);
          string query = "DELETE FROM Student WHERE RoomNumber = '" + roomNumber + "'";

          int rowsAffected = 0;

          connection.Open();
          SqlCommand command = new SqlCommand(query, connection);
          rowsAffected = command.ExecuteNonQuery();
          connection.Close();

          return rowsAffected > 0;
        }
        public  bool UpdateStudentStatus(string username, int roomno, string hostelname, string 
connectionString)
        {
          SqlConnection connection = new SqlConnection(connectionString);
          connection.Open();
          string query = String.Format("UPDATE Student SET StudentStatus = 'Alloted', RoomNumber = 
'{0}', HostelName = '{1}' WHERE Username = '{2}'", roomno, hostelname, username);
```

```
        SqlCommand command = new SqlCommand(query, connection);

        int rowsAffected = command.ExecuteNonQuery();

        connection.Close();


        return rowsAffected > 0;

    }

    public  bool UpdateStudentPassword(string password ,string name, string connectionString)

    {

        SqlConnection connection = new SqlConnection(connectionString);

        connection.Open();

        string query = String.Format("UPDATE Student SET Password = '{0}'  WHERE Username = '{1}'",
password,  name );

        SqlCommand command = new SqlCommand(query, connection);

        int rowsAffected = command.ExecuteNonQuery();

        connection.Close();


        return rowsAffected > 0;

    }

    public  List<Student> GetStudentsWithAllotedStatus(string connectionString)

    {

        List<Student> allotedStudents = new List<Student>();


        SqlConnection connection = new SqlConnection(connectionString);

        connection.Open();


        string query = "SELECT * FROM Student WHERE StudentStatus = 'alloted'";

        SqlCommand command = new SqlCommand(query, connection);


        SqlDataReader reader = command.ExecuteReader();
```

```
            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);
                string password = Convert.ToString(reader["Password"]);
                int age = Convert.ToInt32(reader["Age"]);
                string gender = Convert.ToString(reader["Gender"]);
                string cnic = Convert.ToString(reader["Cnic"]);
                string role = Convert.ToString(reader["Role"]);
                string hostelName = Convert.ToString(reader["HostelName"]);
                string roomNumber = Convert.ToString(reader["RoomNumber"]);
                string studentStatus = Convert.ToString(reader["StudentStatus"]);

                Student student = new Student(username, password, age, cnic, role,
gender,HostelDb.GetHostelByName( hostelName,connectionString),
RoomDb.GetRoomByRoomNumber(roomNumber,connectionString), studentStatus);
                allotedStudents.Add(student);
            }

            reader.Close();
            connection.Close();

            return allotedStudents;
        }
        public  List<Student> GetStudentsByHostelName(string hostelName,string connectionString)
        {
            List<Student> allotedStudents = new List<Student>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();


            string query = string.Format("SELECT * FROM Student WHERE HostelName =
'{0}'",hostelName);
            SqlCommand command = new SqlCommand(query, connection);


            SqlDataReader reader = command.ExecuteReader();


            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);
                string password = Convert.ToString(reader["Password"]);
                int age = Convert.ToInt32(reader["Age"]);
                string gender = Convert.ToString(reader["Gender"]);
                string cnic = Convert.ToString(reader["Cnic"]);
                string role = Convert.ToString(reader["Role"]);
                string hostelname = Convert.ToString(reader["HostelName"]);
                string roomNumber = Convert.ToString(reader["RoomNumber"]);
                string studentStatus = Convert.ToString(reader["StudentStatus"]);


                Student student = new Student(username, password, age, cnic, role, gender,
HostelDb.GetHostelByName(hostelName, connectionString),
RoomDb.GetRoomByRoomNumber(roomNumber, connectionString), studentStatus);
                allotedStudents.Add(student);
            }


            reader.Close();
            connection.Close();
```

```
    return allotedStudents;
}




public List<Student> GetAllStudents(string connectionString)
{
    List<Student> allotedStudents = new List<Student>();

    SqlConnection connection = new SqlConnection(connectionString);
    connection.Open();

    string query = "SELECT * FROM Student";
    SqlCommand command = new SqlCommand(query, connection);

    SqlDataReader reader = command.ExecuteReader();

    while (reader.Read())
    {
        string username = Convert.ToString(reader["Username"]);
        string password = Convert.ToString(reader["Password"]);
        int age = Convert.ToInt32(reader["Age"]);
        string gender = Convert.ToString(reader["Gender"]);
        string cnic = Convert.ToString(reader["Cnic"]);
        string role = Convert.ToString(reader["Role"]);
        string hostelName = Convert.ToString(reader["HostelName"]);
        string roomNumber = Convert.ToString(reader["RoomNumber"]);
        string studentStatus = Convert.ToString(reader["StudentStatus"]);
```

```
            Student student = new Student(username, password, age, cnic, role, gender,
HostelDb.GetHostelByName(hostelName, connectionString),
RoomDb.GetRoomByRoomNumber(roomNumber, connectionString), studentStatus);

            allotedStudents.Add(student);

        }


        reader.Close();

        connection.Close();


        return allotedStudents;

    }


    public  bool DeleteStudent(string studentName, string connectionString)

    {

        SqlConnection connection = new SqlConnection(connectionString);

        string query = string.Format("DELETE FROM Student WHERE Username = '{0}'", studentName);


        int rowsAffected = 0;


        connection.Open();

        SqlCommand command = new SqlCommand(query, connection);

        rowsAffected = command.ExecuteNonQuery();

        connection.Close();


        return rowsAffected > 0;


    }

    public  List<string> GeAllStudentNames(string connectionString)
```

```
{
   List<string> Students = new List<string>();

   SqlConnection connection = new SqlConnection(connectionString);
   connection.Open();

   string query = "SELECT * FROM Student";
   SqlCommand command = new SqlCommand(query, connection);

   SqlDataReader reader = command.ExecuteReader();

   while (reader.Read())
   {
      string username = Convert.ToString(reader["Username"]);



      Students.Add(username);
   }

   reader.Close();
   connection.Close();

   return Students;
}



public  List<string> GetStudentNamesByHostelName(string hostelName, string connectionString)
{
```

```
            List<string> studentNames = new List<string>();


            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT Username FROM Student where Hostelname = '{0}'",
hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();


            while (reader.Read())
            {
                string name = reader.GetString(0);
                studentNames.Add(name);
            }


            reader.Close();
            connection.Close();


            return studentNames;
        }




    }
}
```

1.11.2.2 ResidentDB.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlDb
{
    public class ResidentDB: IResident
    {

        public  bool AddResidents(Resident resident, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("insert into Resident (Username, 
Password, Age,Gender,Cnic,Role) VALUES('{0}', '{1}', '{2}','{3}', '{4}', '{5}')", 
resident.GetName(), resident.GetPassword(), resident.GetAge(), 
resident.GetGender(), resident.GetCnic(), resident.GetRole());
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  bool DeleteResidentByHostelName(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            int rowsAffected = 0;

            if (connection.State != ConnectionState.Open)
                connection.Open();

            string query = string.Format("DELETE FROM Resident WHERE HostelName =
'{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();

            connection.Close();

            return rowsAffected > 0;
        }

        public  Resident IsResidentFound(string name, string password, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("select * from Resident where Username =
'{0}' and Password = '{1}' ", name, password);
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();
            Resident resident;
            if (reader.Read())
            {
                string Username = Convert.ToString(reader["Username"]);
                string Password = Convert.ToString(reader["Password"]);
                string Gender = Convert.ToString(reader["Gender"]);
                int Age = Convert.ToInt32(reader["Age"]);
                string Cnic = Convert.ToString(reader["Cnic"]);
                string Role = Convert.ToString(reader["Role"]);

                resident = new Resident(Username, Password, Age, Cnic, Role,
Gender);
                return resident;
            }
            else return null;
        }
        public  bool UpdateResidentPassword(string password, string name, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Resident SET Password = '{0}'
WHERE Username = '{1}'", password, name);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public  int GetResidentId(string username, string connectionString)
        {
            int residentId = -1; // Default value in case the resident ID is not
found
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("select ResidentId from Resident where 
Username = '{0}'", username);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            if (reader.Read())
            {
                residentId = Convert.ToInt32(reader["ResidentId"]);
            }

            reader.Close();
            connection.Close();

            return residentId;
        }
        public  void UpdateResidentHostelName(string username , string hostelname, 
string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("UPDATE Resident SET HostelName = '{0}' 
WHERE Username = '{1}'", hostelname, username);
            SqlCommand command = new SqlCommand(query, connection);

            command.ExecuteNonQuery();

            connection.Close();
        }

        public  bool DeleteResident(string residentName, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            string query = string.Format("DELETE FROM Resident WHERE Username = 
'{0}'", residentName);


            int rowsAffected = 0;


            connection.Open();

            SqlCommand command = new SqlCommand(query, connection);

            rowsAffected = command.ExecuteNonQuery();

            connection.Close();


            return rowsAffected > 0;


        }


        public  string GetGenderOfSelectedPerson(string username, string 
connectionString)

        {

            string gender = null;

            SqlConnection connection = new SqlConnection(connectionString);

            connection.Open();

            string query = string.Format("SELECT Gender FROM Resident where 
Username = '{0}'", username);

            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();

            if (reader.Read())

            {

                gender = reader.GetString(0);

            }

            reader.Close();

            connection.Close();

            return gender;
        }
    }
}
```

1.11.2.3 RtDb.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;

using ResistayDll.DLInterfaces;

namespace ResistayDll.DlDb
{
    public class RtDb:IRt
    {
        Ihostel HostelDb = new HostelDb();

        public bool DeleteRtByHostelName(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            int rowsAffected = 0;

            if (connection.State != ConnectionState.Open)
                connection.Open();
            string query = string.Format("DELETE FROM Rt WHERE HostelName =
'{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();

            connection.Close();

            return rowsAffected > 0;
        }

        public  bool InsertRt(RT resident ,int residentId,string connectionString)
        {

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("insert into Rt (Username, Password,
Age,Gender,Cnic,Role,PhoneNo,ResidentId,HostelName) VALUES('{0}', '{1}',
'{2}','{3}', '{4}', '{5}','{6}','{7}','{8}')", resident.GetName(),
resident.GetPassword(), resident.GetAge(), resident.GetGender(),
resident.GetCnic(), resident.GetRole(),resident.GetPhoneNumber(),
residentId,resident.GetHostel().GetHostelName());
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  bool DeleteRt(string rtName,string  connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            string query = string.Format("DELETE FROM Rt WHERE Username = 
'{0}'",rtName);

            int rowsAffected = 0;

            connection.Open();
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;

        }

        public  string GetHostelOfSelectedRt(string username, string 
connectionString)
        {
            string hostel = null;
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT HostelName FROM Rt where 
HostelName = '{0}'", username);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                hostel = reader.GetString(0);
            }
            reader.Close();
            connection.Close();
            return hostel;




        }
        public  bool IsDuplicateRt(string username, string connectionString)
        {
            string resident = null;
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT Username FROM Rt WHERE Username = 
'{0}'", username);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                resident = reader.GetString(0);
            }
            reader.Close();
            connection.Close();

            return resident == null;
        }
        public  bool UpdateRtPassword(string password, string name, string 
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Rt SET Password = '{0}'  WHERE 
Username = '{1}'", password, name);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public  RT GetRtByName(string name, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("select * from Rt where Username = '{0}'
", name);
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();
            RT rt;
            if (reader.Read())
            {
                string Username = Convert.ToString(reader["Username"]);
                string Password = Convert.ToString(reader["Password"]);
                string Gender = Convert.ToString(reader["Gender"]);
                int Age = Convert.ToInt32(reader["Age"]);
                string Cnic = Convert.ToString(reader["Cnic"]);
                string hostelName = Convert.ToString(reader["HostelName"]);

                string phoneNo = Convert.ToString(reader["PhoneNo"]);
                string Role = Convert.ToString(reader["Role"]);

                rt = new RT(Username, Password, Age, Cnic, Role, Gender,
HostelDb.GetHostelByName(hostelName, connectionString),phoneNo);
                return rt;
            }
            else return null;
        }
        public  List<RT> GeAllRts (string connectionString)
        {
            List<RT> Rts = new List<RT>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = "SELECT * FROM Rt";
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);
                string password = Convert.ToString(reader["Password"]);
                int age = Convert.ToInt32(reader["Age"]);
                string gender = Convert.ToString(reader["Gender"]);
                string cnic = Convert.ToString(reader["Cnic"]);
                string role = Convert.ToString(reader["Role"]);
                string hostelName = Convert.ToString(reader["HostelName"]);
                string phoneNumber = Convert.ToString(reader["PhoneNo"]);



                RT rt = new RT(username, password, age, cnic, role, gender,
HostelDb.GetHostelByName(hostelName, connectionString), phoneNumber);
                Rts.Add(rt);
            }

            reader.Close();
            connection.Close();
            return Rts;
        }
        public List<string> GeAllRtNames(string connectionString)
        {
            List<string> Rts = new List<string>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = "SELECT * FROM Rt";
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string username = Convert.ToString(reader["Username"]);



                 Rts.Add(username);
            }

            reader.Close();
            connection.Close();

            return Rts;
        }

    }
}
```

1.11.2.4 RoomDb.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Net.NetworkInformation;

using System.Text;

using System.Threading.Tasks;


using ResistayDll.DLInterfaces;


namespace ResistayDll.DlDb

{

    public class RoomDb:IRoom

    {

        Ihostel HostelDb = new HostelDb();

        public  bool InsertRoom(Room room, string hostelName, string 
connectionString)

        {

            SqlConnection connection = new SqlConnection(connectionString);


            string query = string.Format("INSERT INTO Room (RoomNumber, 
HostelName, RoomStatus) VALUES ('{0}', '{1}', '{2}')",

                                          room.GetNumber(), hostelName, 
room.GetRoomStatus());


            SqlCommand command = new SqlCommand(query, connection);


            int rowsAffected = 0;


            connection.Open();
            rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }

        public  List<int> GetRoomNumbersByHostelName(string hostelName, string 
connectionString)
        {
            List<int> roomNumbers = new List<int>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format( "SELECT RoomNumber FROM Room WHERE 
HostelName = '{0}'and RoomStatus ='Vacant'",hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@HostelName", hostelName);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int roomNumber = Convert.ToInt32(reader["RoomNumber"]);
                roomNumbers.Add(roomNumber);
            }

            reader.Close();
            connection.Close();

            return roomNumbers;
        }


        public  List<Room> GetAvailableRooms(string connectionString)
        {
            List<Room> availableRooms = new List<Room>();

            SqlConnection connection = new SqlConnection(connectionString);
            SqlCommand command = connection.CreateCommand();
            command.CommandText = "SELECT * FROM Room WHERE RoomStatus =
'Vacant'";

            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int roomNumber = Convert.ToInt32(reader["RoomNumber"]);

                string roomStatus = Convert.ToString(reader["RoomStatus"]);
                string HostelName = Convert.ToString(reader["HostelName"]);

                Room room = new Room(roomNumber,
roomStatus,HostelDb.GetHostelByName(HostelName,connectionString));
                availableRooms.Add(room);
            }

            reader.Close();
            connection.Close();

            return availableRooms;
        }

        public  bool DeleteRoomByHostelName(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            int rowsAffected = 0;

            if (connection.State != ConnectionState.Open)
                connection.Open();

            string query = string.Format("DELETE FROM Room WHERE HostelName = 
'{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();

            connection.Close();

            return rowsAffected > 0;
        }
        public  Room GetRoomByRoomNumber(string roomNumber, string 
connectionString)
        {
            Room room = null;

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("SELECT * FROM Room WHERE RoomNumber = 
'{0}'", roomNumber);
            SqlCommand command = new SqlCommand(query, connection);

            SqlDataReader reader = command.ExecuteReader();

            if (reader.Read())
            {
                int number = Convert.ToInt32(reader["RoomNumber"]);
                string status = Convert.ToString(reader["RoomStatus"]);
                // Add more fields as per your Room class properties
                room = new Room(number, status);
            }

            reader.Close();
            connection.Close();

            return room;
        }

        public  bool DeleteRoomByRoomNumber(int roomNumber,string hostel, string 
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            string query = string.Format("DELETE FROM Room WHERE RoomNumber = 
'{0}' AND HostelName = '{1}'",roomNumber,hostel);

            int rowsAffected = 0;

            connection.Open();
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public  List<int> GetRoomNumbers(string connectionString)
        {
            List<int> roomNumbers = new List<int>();

            SqlConnection connection = new SqlConnection(connectionString);
            string query = "SELECT RoomNumber FROM Room";
            SqlCommand command = new SqlCommand(query, connection);

            connection.Open();
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int roomNumber = reader.GetInt32(0);
                roomNumbers.Add(roomNumber);
            }

            reader.Close();
            connection.Close();

            roomNumbers.Distinct().ToList();
            return roomNumbers;
        }

        public  bool UpdateRoomStatusVacant(int roomno, string hostel,string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Room SET RoomStatus = 'Vacant'
WHERE RoomNumber = '{0}'and HostelName = '{1}'", roomno,hostel );
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public  bool UpdateRoomStatusAlloted(int roomno, string hostel, string
connectionString)
```

```
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Room SET RoomStatus = 'Alloted' 
WHERE RoomNumber = '{0}'and HostelName = '{1}'", roomno,hostel);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public bool IsDuplicateRoom(string hostelName, int roomNo, string 
connectionString)
        {
            int foundRoomNo = -1; // Variable to store the found room number
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT RoomNumber FROM Room WHERE 
HostelName = '{0}' AND RoomNumber = '{1}'", hostelName, roomNo);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                foundRoomNo = reader.GetInt32(0); // Assign the found room number
            }
            reader.Close();
            connection.Close();

            // Check if a room with the given hostel name and room number exists
            return foundRoomNo == -1;
        }

    }
```

```
}
```

1.11.2.5 HostelDb.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;
namespace ResistayDll.DlDb
{
    public class HostelDb : Ihostel
    {
        private static string query;

        public  bool InsertHostel(Hostel hostel, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);

            connection.Open();

            string query = string.Format("INSERT INTO Hostel (HostelName,
HostelType, HostelStatus) VALUES ('{0}', '{1}', '{2}')",
                                                             hostel.GetHostelName(),
hostel.GetHostelType(), hostel.GetHostelStatus());

            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }


        }
        public  bool DeleteHostelByHostelName(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            int rowsAffected = 0;

            if (connection.State != ConnectionState.Open)
                connection.Open();

            string query = string.Format("DELETE FROM Hostel WHERE HostelName =
'{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            rowsAffected = command.ExecuteNonQuery();

            connection.Close();

            return rowsAffected > 0;
        }
        public  List<Hostel> GetAllHostels(string connectionString)
        {
            List<Hostel> hostels = new List<Hostel>();
            SqlConnection connection = new SqlConnection(connectionString);


            connection.Open();
            string query = "SELECT * FROM Hostel";


            SqlCommand command = new SqlCommand(query, connection);


            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string hostelName =
reader.GetString(reader.GetOrdinal("HostelName"));
                    string hostelType =
reader.GetString(reader.GetOrdinal("HostelType"));
                    string hostelStatus =
reader.GetString(reader.GetOrdinal("HostelStatus"));


                    Hostel hostel = new Hostel(hostelName, hostelType,
hostelStatus);
                    hostels.Add(hostel);
                }
            }




            return hostels;
        }


        public  List<string> GetHostelNames(string connectionString)
        {
            List<string> Hostels = new List<string>();


            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = ""; // Declare the query variable here

            query = "SELECT HostelName FROM Hostel ";




            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string hostel = Convert.ToString(reader["HostelName"]);
                Hostels.Add(hostel);
            }

            reader.Close();
            connection.Close();

            return Hostels;
        }

        public  List<string> GetHostelNamesByGender(string gender, string
connectionString)
        {
            List<string> AvailableHostels = new List<string>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = ""; // Declare the query variable here

            if (gender == "male")
            {
                query = "SELECT HostelName FROM Hostel where HostelType = 'boys'";
            }
            else
            {
                query = "SELECT HostelName FROM Hostel where HostelType =
'girls'";
            }

            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string hostel = Convert.ToString(reader["HostelName"]);
                AvailableHostels.Add(hostel);
            }

            reader.Close();
            connection.Close();

            return AvailableHostels;
        }
        public  Hostel GetHostelByName(string hostelName, string connectionString)
        {
            Hostel hostel = null;

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();

            string query = string.Format("SELECT * FROM Hostel WHERE HostelName =
'{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            if (reader.Read())
            {
                string name = Convert.ToString(reader["HostelName"]);
                string type = Convert.ToString(reader["HostelType"]);
                string status = Convert.ToString(reader["HostelStatus"]);
                // Add more fields as per your Hostel class properties

                hostel = new Hostel(name, type, status);
            }

            reader.Close();
            connection.Close();

            return hostel;
        }
        public  List<string> GetHostelBySelectedddltRoom(int room, string
connectionString)
        {
            List<string> Hostels = new List<string>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = ""; // Declare the query variable here

            query = string.Format("SELECT HostelName FROM Room where RoomNumber =
'{0}' ", room);



            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                string hostel = Convert.ToString(reader["HostelName"]);
                Hostels.Add(hostel);
            }

            reader.Close();
            connection.Close();

            return Hostels;

        }

        public  bool IsDuplicateHostel(string hostelName, string connectionString)
        {
            string hostel = null;
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT HostelName FROM Hostel WHERE
HostelName = '{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                hostel = reader.GetString(0);
            }
            reader.Close();
            connection.Close();

            return hostel == null;
        }


        public bool UpdateHostelStatusUnchecked(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Hostel SET HostelStatus =
'Unchecked' WHERE HostelName = '{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
        public  bool UpdateHostelStatusChecked(string hostelName, string
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("UPDATE Hostel SET HostelStatus =
'Checked' WHERE HostelName = '{0}'", hostelName);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();

            return rowsAffected > 0;
        }
    }
}
```

1.11.2.6 ComplainDb.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlDb
{
    public class ComplainDb: IComplain
    {

        public  bool AddComplain( Complain complain,string username, string 
connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("insert into Complain 
(Detail,Date,StudentName) VALUES('{0}','{1}','{2}' )", complain.GetDescription(), 
complain.GetDate(),username);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  bool DeleteComplain(int complainId, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("DELETE FROM Complain WHERE ComplainId =
{0}", complainId);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  List<Complain> GetAllComplainsofStudent(string studentname, string
connectionString)
        {
            List<Complain> complains = new List<Complain>();

            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string query = String.Format("SELECT * FROM Complain where
StudentName = '{0}'",studentname);
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    string description = reader["Detail"].ToString();
                    string date = reader["date"].ToString(); // Assuming 'date' is
stored as string in the database
                    Complain complain = new Complain(description, date);
                    complains.Add(complain);
                }

                reader.Close();
            }

            return complains;
        }

        public  List<int> GetComplainId(string studentName,string
connectionString)
        {
            List<int> complainIds = new List<int>();

            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = string.Format("SELECT ComplainId FROM Complain where
StudentName= '{0}'",studentName);
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int complainId = Convert.ToInt32(reader["ComplainId"]);
                 complainIds.Add(complainId);
            }

            reader.Close();
            connection.Close();
            return complainIds;
        }
        public  bool DeleteComplainbyStudentName(string studentName, string connectionString)
        {
            SqlConnection connection = new SqlConnection(connectionString);
            connection.Open();
            string query = String.Format("DELETE FROM Complain WHERE StudentName = '{0}'", studentName);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

1.11.2.7 AnnoucementDb.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlDb
{
    public class AnnoucementDb: IAnnoucement
    {
        private string ConnectionString = "";
        public AnnoucementDb(string ConnectionString)
        {
            this.ConnectionString = ConnectionString;
        }
        public  bool AddAnnoucement(Annoucement annoucement)
        {
            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = String.Format("insert into Annoucement (Details,Date)
VALUES('{0}','{1}' )", annoucement.GetDescription(),annoucement.GetDate());
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  bool DeleteAnnoucement(int announcementId )
        {
            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = String.Format("DELETE FROM Annoucement WHERE
AnnoucemntId = {0}", announcementId);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public  List<Annoucement> GetAllAnnouncements( )
        {
            List<Annoucement> announcements = new List<Annoucement>();

            SqlConnection connection = new SqlConnection(ConnectionString);

                connection.Open();
                string query = "SELECT * FROM Annoucement";
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    string description = reader["Details"].ToString();
                    string date = reader["date"].ToString();
                int id = Convert.ToInt32(reader["AnnoucemntId"]);
                Annoucement announcement = new Annoucement(description,date,id);
                    announcements.Add(announcement);
                }

                reader.Close();



            return announcements;
        }
        public  List<int> GetAnnoucementId( )
        {
            List<int> announcementIds = new List<int>();

            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = "SELECT AnnoucemntId FROM Annoucement";
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
            {
                int announcementId = Convert.ToInt32(reader["AnnoucemntId"]);
                announcementIds.Add(announcementId);
            }

            reader.Close();
            connection.Close();

            return announcementIds;
        }
    }
}
```

1.11.2.8 RuleDb.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlDb
{
    public class RulesDb:IRule
    {
        private string ConnectionString = "";
        public RulesDb(string ConnectionString)
        {
            this.ConnectionString = ConnectionString;
        }
        public  bool AddRule(Rule rule)
        {
            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = String.Format("insert into Rules (Description, Date)
VALUES('{0}', '{1}')", rule.GetDescription(), rule.GetDate());
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public bool DeleteRule(int ruleId )
        {
            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = String.Format("DELETE FROM Rules WHERE rulesId = {0}", 
ruleId);
            SqlCommand command = new SqlCommand(query, connection);
            int rowsAffected = command.ExecuteNonQuery();
            connection.Close();
            if (rowsAffected > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
        public List<Rule> GetAllRules( )
        {
            List<Rule> rules = new List<Rule>();
            using (SqlConnection connection = new SqlConnection(ConnectionString))
            {
                connection.Open();
                string query = "SELECT * FROM Rules ";
                SqlCommand command = new SqlCommand(query, connection);
                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())
                {
                    string description = reader["description"].ToString();
                    string date = reader["date"].ToString();
                    Rule rule = new Rule(description, date);
                    rules.Add(rule);
                }

                reader.Close();
            }

            return rules;
        }
        public List<int> GetRuleIds()
        {
            List<int> ruleIds = new List<int>();

            SqlConnection connection = new SqlConnection(ConnectionString);
            connection.Open();
            string query = "SELECT RulesId FROM Rules";
            SqlCommand command = new SqlCommand(query, connection);
            SqlDataReader reader = command.ExecuteReader();

            while (reader.Read())
```

```
                {
                    int ruleId = Convert.ToInt32(reader["RulesId"]);
                    ruleIds.Add(ruleId);
                }

                reader.Close();
                connection.Close();

                return ruleIds;
            }
        }
    }
```

1.11.3 DL Folder with FileHandling

1.11.3.1 AnnoucementFH.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlFH
{
    public class AnnoucementFH : IAnnoucement
    {
        private string Path;
        private int Id = 0;
```

```
public AnnoucementFH(string Path)
{
    this.Path = Path;
    LoadAnnoucementFromFile();
}
private List<Annoucement> Annoucements = new List<Annoucement>();

private bool AddAnnoucementInList(Annoucement annoucement)
{
    if (annoucement != null)
    {
        Annoucements.Add(annoucement);
        return true;
    }
    else
    {
        return false;
    }
}
private void LoadAnnoucementFromFile()
{
    string record;
    StreamReader streamReader = new StreamReader(Path);
    List<Annoucement> annoucements = new List<Annoucement>();
    if(File.Exists(Path))
    {
        while((record =streamReader.ReadLine())!=null)
        {
            string[] splitrecord = record.Split(',');
            int annoucementId = Convert.ToInt32(splitrecord[0]);
            string annoucementDetail = splitrecord[1];
            string annoucementDate = splitrecord[2];
```

```
                    Annoucement annoucement = new
Annoucement(annoucementDetail,annoucementDate,annoucementId);
                    Id++;
                    AddAnnoucementInList(annoucement);
                }
                streamReader.Close();
            }
        }
        public List<Annoucement> GetAllAnnouncements()
        {
            return Annoucements;
        }
        public bool AddAnnoucement(Annoucement annoucement)
        {
            bool Check = false;
            Id++;
            annoucement.SetId(Id);
            StreamWriter streamWriter = new StreamWriter(Path,false);
            if(AddAnnoucementInList(annoucement)==true)
            {
                Check = true;
            }
            foreach(Annoucement Annoucement in Annoucements)
            {
                streamWriter.WriteLine(Annoucement.GetId() + "," +
Annoucement.GetDescription() + "," + Annoucement.GetDate());
                streamWriter.Flush();
            }
            streamWriter.Close();
            return Check;
        }
        public List<int> GetAnnoucementId()
        {
            List<int> announcementIds = new List<int>();
            foreach (Annoucement annoucement in Annoucements)
            {
                announcementIds.Add(annoucement.GetId());
            }
            return announcementIds;
        }
        public bool DeleteAnnoucement(int announcementId)
        {
            foreach (Annoucement annoucement in Annoucements)
            {
                if (annoucement.GetId() == announcementId)
                {
                    Annoucements.Remove(annoucement);
                    return true;
                }
            }
            return false;
        }


    }
}
```

1.11.3.2 RuleFH.cs

```
using ResistayDLL.Bl;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DLInterfaces;

namespace ResistayDll.DlFH
{
    public class RulesFH : IRule
    {
        private string Path;
        private int Id = 0;
        public RulesFH(string Path)
        {
            this.Path = Path;
            LoadRulesFromFile();
        }
        private List<Rule>  Rules = new List<Rule>();

        private bool AddRuleInList(Rule rule)
        {
            if (rule != null)
            {
                Rules.Add(rule);
                return true;
            }
            else
            {
                return false;
            }
        }
        private void LoadRulesFromFile()
        {
            string record;
            StreamReader streamReader = new StreamReader(Path);
            List<Rule> ruless = new List<Rule>();
            if (File.Exists(Path))
            {
                while ((record = streamReader.ReadLine()) != null)
                {
                    string[] splitrecord = record.Split('|');
                    int ruleId = Convert.ToInt32(splitrecord[0]);
                    string ruleDetail = splitrecord[1];
                    string ruleDate = splitrecord[2];
                    Rule rule = new Rule (ruleDetail, ruleDate, ruleId);
                    Id++;
                    AddRuleInList(rule);
                }
                streamReader.Close();
            }
        }
        public List<Rule> GetAllRules()
        {
            return Rules;
        }
        public bool AddRule(Rule rule)
        {
            bool Check = false;
            Id++;
            rule.SetId(Id);
            StreamWriter streamWriter = new StreamWriter(Path, false);
            if (AddRuleInList(rule) == true)
            {
                Check = true;
            }
            foreach (Rule Rule in Rules)
            {
                streamWriter.WriteLine(Rule.GetId() + "|" + Rule.GetDescription()
 + "|" + Rule.GetDate());
                streamWriter.Flush();
            }
            streamWriter.Close();
            return Check;
        }
        public List<int> GetRuleIds()
        {
            List<int> ruleIds = new List<int>();
            foreach (Rule Rule in Rules)
            {
                ruleIds.Add(Rule.GetId());
            }
            return ruleIds;
        }
        public bool DeleteRule(int ruleId)
        {
            foreach (Rule Rule in Rules)
            {
                if (Rule.GetId() == ruleId)
                {
                    Rules.Remove(Rule);
                    Id--;
                    UpdateFile();
                    return true;
                }
            }
            return false;
        }
        private void UpdateFile()
        {
            StreamWriter streamWriter = new StreamWriter(Path, false);
            foreach (Rule rule in Rules)
            {
                streamWriter.WriteLine(rule.GetId() + "|" + rule.GetDescription()
+ "|" + rule.GetDate());
            }
            streamWriter.Close();
        }
    }
}
```

1.11.4 DL Interfaces Folder

1.11.4.1 IStudent.cs

```
using ResistayDLL.Bl;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDll.DLInterfaces
{
    public interface IStudent
    {
        bool AddStudent(Student resident, int ResidentId, string
connectionString);
        bool DeleteStudentByHostelName(string hostelName, string
connectionString);
        List<string> GetNameOfUnallotedStudents(string connectionString);
        string GetHostelOfSelectedStudent(string username, string
connectionString);
        bool IsDuplicateStudent(string username, string connectionString);
        Student GetStudentByName(string name, string connectionString);

        int GetRoomNoOfSelectedStudent(string studentName, string
connectionString);

        bool DeleteStudentByRoomNumber(int roomNumber, string connectionString);

        bool UpdateStudentStatus(string username, int roomno, string hostelname,
string connectionString);

        bool UpdateStudentPassword(string password, string name, string
connectionString);

        List<Student> GetStudentsWithAllotedStatus(string connectionString);

        List<Student> GetStudentsByHostelName(string hostelName, string
connectionString);

        List<Student> GetAllStudents(string connectionString);

        bool DeleteStudent(string studentName, string connectionString);

        List<string> GeAllStudentNames(string connectionString);

        List<string> GetStudentNamesByHostelName(string hostelName, string
connectionString);

    }

}
```

1.11.4.2 IResident.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace ResistayDll.DLInterfaces

{

    public interface IResident

    {

        bool AddResidents(Resident resident, string connectionString);
        bool DeleteResidentByHostelName(string hostelName, string connectionString);

        Resident IsResidentFound(string name, string password, string connectionString);

        bool UpdateResidentPassword(string password, string name, string connectionString);

        int GetResidentId(string username, string connectionString);

        void UpdateResidentHostelName(string username, string hostelname, string connectionString);

        bool DeleteResident(string residentName, string connectionString);

        string GetGenderOfSelectedPerson(string username, string connectionString);

    }

}
```

1.11.4.3 IRt.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace ResistayDll.DLInterfaces

{

    public interface IRt

    {

        bool DeleteRtByHostelName(string hostelName, string connectionString);

        bool InsertRt(RT resident, int residentId, string connectionString);

        bool DeleteRt(string rtName, string connectionString);

        string GetHostelOfSelectedRt(string username, string connectionString);

        bool IsDuplicateRt(string username, string connectionString);
        bool UpdateRtPassword(string password, string name, string
connectionString);

        RT GetRtByName(string name, string connectionString);

        List<RT> GeAllRts(string connectionString);

        List<string> GeAllRtNames(string connectionString);

    }

}
```

1.11.4.4 IRoom.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace ResistayDll.DLInterfaces

{

    public interface IRoom

    {

        bool InsertRoom(Room room, string hostelName, string connectionString);

        List<int> GetRoomNumbersByHostelName(string hostelName, string
connectionString);

        List<Room> GetAvailableRooms(string connectionString);

        bool DeleteRoomByHostelName(string hostelName, string connectionString);

        Room GetRoomByRoomNumber(string roomNumber, string connectionString);

        bool DeleteRoomByRoomNumber(int roomNumber, string hostel, string
connectionString);

        List<int> GetRoomNumbers(string connectionString);

        bool UpdateRoomStatusVacant(int roomno, string hostel, string
connectionString);

        bool UpdateRoomStatusAlloted(int roomno, string hostel, string
connectionString);
```

```
        bool IsDuplicateRoom(string hostelName, int roomNo, string
connectionString);
    }
}
```

1.11.4.5 IHostel.cs

```
using ResistayDLL.Bl;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ResistayDll.DLInterfaces
{
    public interface Ihostel
    {
        bool InsertHostel(Hostel hostel, string connectionString);
        bool DeleteHostelByHostelName(string hostelName, string connectionString);
        List<Hostel> GetAllHostels(string connectionString);
        List<string> GetHostelNames(string connectionString);
        List<string> GetHostelNamesByGender(string gender, string
connectionString);
        Hostel GetHostelByName(string hostelName, string connectionString);
        List<string> GetHostelBySelectedddltRoom(int room, string
connectionString);
        bool IsDuplicateHostel(string hostelName, string connectionString);
        bool UpdateHostelStatusUnchecked(string hostelName, string
connectionString);
        bool UpdateHostelStatusChecked(string hostelName, string
connectionString);
    }
}
```

1.11.4.6 IComplain.cs

```
using ResistayDLL.Bl;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace ResistayDll.DLInterfaces

{

    public interface IComplain

    {

        bool AddComplain(Complain complain, string username, string
connectionString);

        bool DeleteComplain(int complainId, string connectionString);

        List<Complain> GetAllComplainsofStudent(string studentname, string
connectionString);

        List<int> GetComplainId(string studentName, string connectionString);

        bool DeleteComplainbyStudentName(string studentName, string
connectionString);


    }

}
```

1.11.4.7 IAnnoucement.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;
using ResistayDLL.Bl;

namespace ResistayDll.DLInterfaces
{
     public interface IAnnoucement
    {
        bool AddAnnoucement(Annoucement annoucement);
        bool DeleteAnnoucement(int announcementId);
        List<Annoucement> GetAllAnnouncements();
        List<int> GetAnnoucementId();
    }
}
```

1.11.4.8 IRule.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;

namespace ResistayDll.DLInterfaces
{
    public interface IRule
    {
        bool AddRule(Rule rule);
        bool DeleteRule(int ruleId);
        List<Rule> GetAllRules();
        List<int> GetRuleIds();
    }
}
```

1.11.5 Utilities Folder

1.11.5.1 Validation.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace ResistayDLL.Utilities

{

    public static class Validation

    {

        public static bool IsEmptyInput(string input)

        {

            return !string.IsNullOrEmpty(input);

        }


        public static bool IsValidUsername(string username)

        {

            return IsEmptyInput(username)  ;

        }


        public static bool IsValidPassword(string password)

        {

            return IsEmptyInput(password) && password.Length >= 8 &&
password.Any(char.IsUpper) && password.Any(char.IsLower) &&
password.Any(char.IsDigit);

        }


        public static bool IsValidGender(string gender)

        {
            return IsEmptyInput(gender) && (gender.ToLower() == "male" ||
gender.ToLower() == "female");
        }




        public static bool IsValidCNIC(string cnic)
        {
            return IsEmptyInput(cnic) && cnic.StartsWith("33") && cnic.Length ==
13 && cnic.Substring(2).All(char.IsDigit);
        }

        public static bool IsValidAge(string age)
        {
            if (int.TryParse(age, out int ageValue))
            {
                if (ageValue > 0)
                {
                    return true; // Age is a positive integer greater than 0
                }
            }
            return false;
        }
        public static bool IsValidRoom(string room)
        {
            if (int.TryParse(room, out int roomValue))
            {
                if (roomValue > 0)
                {
                    return true; // Age is a positive integer greater than 0
                }
            }
            return false;
        }


        public static bool IsValidRole(string role)
        {
            return IsEmptyInput(role) && (role.ToLower() == "student");
        }
        public static bool IsValidPhoneNumber(string phoneNumber)
        {
            return IsEmptyInput(phoneNumber) && phoneNumber.StartsWith("03") &&
phoneNumber.Length == 11 && phoneNumber.Substring(2).All(char.IsDigit);
        }
    }
}
```

1.11.6 UI Folder with Console

1.11.6.1 MainMenu.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace ResistayConsole.UI
{
    public class MainMenu
    {
        public static int ShowMainMenu()
        {
            int option = 0;
            while ( option < 1 || option > 3)
            {
                Console.WriteLine("Annoucements");
                Console.WriteLine("Rules");
                Console.WriteLine("Exit");
                option = int.Parse(Console.ReadLine());

            }
            return option;

        }
    }
}
```

1.11.6.2 AnnoucementUI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DlDb;
using ResistayDll.DlFH;
using ResistayDll.DLInterfaces;

namespace ResistayConsole.UI
{
    public class AnnoucementUI
    {
        //private string ConnectionString = @"Data Source=DESKTOP-
KOU2P2U\SQLEXPRESS01;Initial Catalog=Resistay;Integrated Security=True;";
        private string Path = "D:\\Computer science\\OOP\\Business Application
Project\\Project\\TextFiles\\Annoucement.txt";
        IAnnoucement Annoucement;
        public AnnoucementUI()
```

```
            {
            this.Annoucement = new AnnoucementFH(Path);
            }

        public void ShowAnnouncementMenu()
        {
            bool exit = false;
            while (!exit)
            {
                Console.WriteLine("1. Add Announcement");
                Console.WriteLine("2. Delete Announcement");
                Console.WriteLine("3. Show All Announcements");
                Console.WriteLine("4. Exit");
                Console.Write("Choose an option: ");

                string choice = Console.ReadLine();
                switch (choice)
                {
                    case "1":
                        AddAnnouncement();
                        break;
                    case "2":
                        DeleteAnnouncement();
                        break;
                    case "3":
                        ShowAllAnnouncements();
                        break;
                    case "4":
                        exit = true;
                        break;
                    default:
                        Console.WriteLine("Invalid option. Please choose again.");
                    break;
                }
            }
        }

        private void AddAnnouncement()
        {
            Console.Write("Enter announcement description: ");
            string description = Console.ReadLine();
            Console.Write("Enter announcement date: ");
            string date = Console.ReadLine();

            // Add validation as per your requirements

            Annoucement.AddAnnoucement(new Annoucement(description, date));
            Console.WriteLine("Announcement added successfully.");
        }

        private void DeleteAnnouncement()
        {
            Console.Write("Enter announcement ID to delete: ");
            if (int.TryParse(Console.ReadLine(), out int id))
            {
                Annoucement.DeleteAnnoucement(id);
                Console.WriteLine("Announcement deleted successfully.");
            }
            else
            {
                Console.WriteLine("Invalid ID.");
            }
        }
        private void ShowAllAnnouncements()
        {
            var announcements = Annoucement.GetAllAnnouncements();
            foreach (var announcement in announcements)
            {
                Console.WriteLine($"ID: {announcement.GetId()}, Description: 
{announcement.GetDescription()}, Date: {announcement.GetDate()}");
            }
        }

    }
}
```

1.11.6.3 RuleUI.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using ResistayDLL.Bl;
using ResistayDll.DlDb;
using ResistayDll.DlFH;
using ResistayDll.DLInterfaces;

namespace ResistayConsole.UI
{
    public class RulesUI
    {
        //private string ConnectionString = @"Data Source=DESKTOP-
KOU2P2U\SQLEXPRESS01;Initial Catalog=Resistay;Integrated Security=True;";
```

```
        private string Path = "D:\\Computer science\\OOP\\Business Application
Project\\Project\\TextFiles\\Rules.txt";

        IRule Rules;

        public RulesUI()

        {

            this.Rules = new RulesFH(Path);

        }


        public void RulesMenu()

        {

            bool exit = false;

            while (!exit)

            {

                Console.WriteLine("1. Add Rule");

                Console.WriteLine("2. Delete Rule");

                Console.WriteLine("3. Show All Rules");

                Console.WriteLine("4. Exit");

                Console.Write("Choose an option: ");


                string choice = Console.ReadLine();

                switch (choice)

                {

                    case "1":

                        AddRule();

                        break;

                    case "2":

                        DeleteRule();

                        break;

                    case "3":

                        ShowAllRules();

                        break;

                    case "4":
```

```
                    exit = true;
                    break;
                default:
                    Console.WriteLine("Invalid option. Please choose again.");
                    break;
            }
        }
    }

    private void AddRule()
    {
        Console.Write("Enter rule description: ");
        string description = Console.ReadLine();
        Console.Write("Enter rule date: ");
        string date = Console.ReadLine();

        // Add validation as per your requirements

        Rules.AddRule(new Rule(description, date));
        Console.WriteLine("Rule added successfully.");
    }

    private void DeleteRule()
    {
        Console.Write("Enter rule ID to delete: ");
        if (int.TryParse(Console.ReadLine(), out int id))
        {
            Rules.DeleteRule (id);
            Console.WriteLine("Rule deleted successfully.");
        }
        else
        {
                Console.WriteLine("Invalid ID.");
            }
        }

        private void ShowAllRules()
        {
            var announcements = Rules.GetAllRules();
            foreach (var rule in announcements)
            {
                Console.WriteLine($"ID: {rule.GetId()}, Description: 
{rule.GetDescription()}, Date: {rule.GetDate()}");
            }
        }

    }
}
```

1.12 Future Direction

I want to Implement an intelligent room allotment system that considers various factors such as student preferences, compatibility, and proximity to common facilities. This can enhance the overall satisfaction of residents. I can Foster a sense of community by incorporating features that encourage social interactions among hostel residents. This could include discussion forums, event planning, and group messaging to facilitate communication and collaboration.